Bringing Network Coding into SDN: A Case-study for Highly Meshed Heterogeneous Communications

Alejandro Cohen, Homa Esfahanizadeh, Bruno Sousa, João P. Vilela, Miguel Luis, Duarte Raposo, François Michel, Susana Sargento, and Muriel Médard

Abstract

Modern communications have moved away from point-to-point models to increasingly heterogeneous network models. In this article, we propose a novel controller-based protocol to deploy adaptive causal network coding in heterogeneous and highly-meshed communication networks. Specifically, we consider using Software-Defined-Network (SDN) as the main controller. We first present an architecture for the highly-meshed heterogeneous multi-source multi-destination networks that represents the practical communication networks encountered in the fifth generation of wireless networks (5G) and beyond. Next, we present a promising solution to deploy network coding over the new architecture. In fact, we investigate how to generalize adaptive and causal random linear network coding (AC-RLNC), proposed for multipath multi-hop (MP-MH) communication channels, to a protocol for the new multi-source multi-destination network architecture using controller. To this end, we present a modularized implementation of AC-RLNC solution where the modules work together in a distributed fashion and perform the AC-RLNC technology. We also present a new controller-based setting through which the network coding modules can communicate and can attain their required information. Finally, we briefly discuss how the proposed architecture and network coding solution provide a good opportunity for future technologies, e.g., distributed coded computation and storage, mmWave communication environments, and innovative and efficient security features.

I. Introduction

The increasing demand for network connectivity and high data rates requires the efficient utilization of all possible resources. In recent years, the connectivity moved forward from point-to-point schemes to multi-source multi-destination (MS-MD) meshed network of interconnected nodes in which intermediate nodes can cooperate and share the physical resources for efficient and reliable communications.

The current state-of-the-art research lacks a comprehensive understanding of the interplay among coding and scheduling within a heterogeneous highly meshed MS-MD network context. To embrace the groundbreaking 5G network at massive scale in their dense environments, we present efficient adaptive and causal random linear network coding (AC-RLNC) techniques to provide robustness to service degradation and provide effective and reliable network communications. Furthermore, we will exploit the information that can be provided by the Software-Defined-Network (SDN), [1]. In the adaptive network coding, we suggest learning, in real time, the current erasure pattern and the network link rates, using a joint control, to consequently improve the decisions.

The proposed network-coding based solution offer benefits for ultra-reliable communication for highly-interconnected MS-MD networks. As a promising example, the communication solution presented in this article can be very beneficial for realizing the advanced requirements of Smart Cities. One of the main priorities consists on the integration of heterogeneous communication infrastructures and processes to enable the right environment where digital networks and services can prosper, see Fig. 1.

A. Cohen, H. Esfahanizadeh, and M. Médard are with the Research Laboratory of Electronics, MIT, Cambridge, MA, USA (e-mail:cohenale@mit.edu; homaesf@mit.edu; medard@mit.edu). B. Sousa is with the Department of Informatics Engineering of the University of Coimbra, Portugal (e-mail:bmsousa@dei.uc.pt). J. P. Vilela is with CRACS/INESCTEC and the Department of Computer Science of the Faculty of Sciences of the University of Porto, Portugal (e-mail:jvilela@fc.up.pt). M. Luís is with Instituto Superior de Engenharia de Lisboa and Instituto de Telecomunicações, Portugal (e-mail:nmal@av.it.pt). D. Raposo is with Instituto de Telecomunicações, Portugal (e-mail:dmgraposo@av.it.pt). F. Michel is with UCLouvain, Belgium (e-mail:francois.michel@uclouvain.be). S. Sargento is with the University of Aveiro and Instituto de Telecomunicações, Portugal (e-mail:susana@ua.pt).

Part of this work is submitted to IEEE Communications Magazine on Transport Layer Innovations for Future Networks.

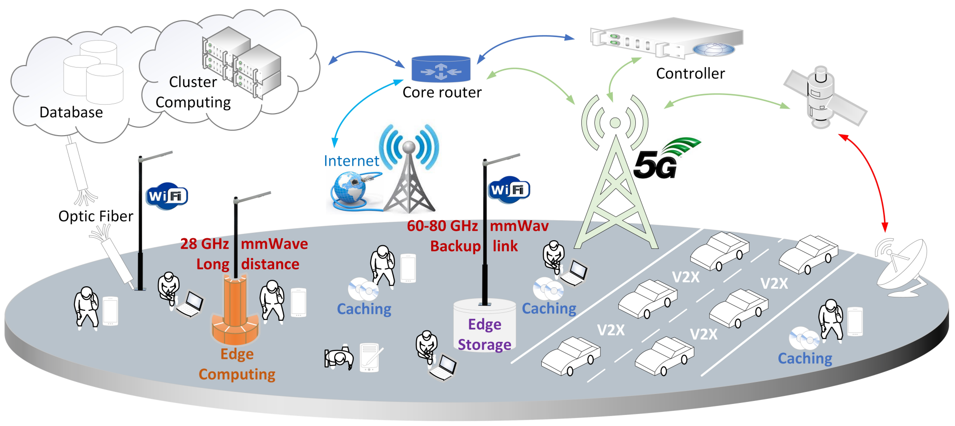

Fig. 1: Heterogeneity in meshed communications.

Upcoming communication networks, 5G and beyond, will be faced with stringent capacity and latency requirements. With an impressive growth of new services and devices (consuming and producing huge amounts of data), mobile networks must become ready to grant the expected quality of service to its users. However, achieving the 5G key performance indicators especially in highly dense networks of devices, such as urban areas of the major cities, will require complementary solutions. For example, to cope with the lack of connectivity, or simply to increase the backhaul link capacity traditionally supported by fiber links, high frequency and high capacity millimeter-wave (mmWave) communications have proven to be an excellent candidate [2]. In such situations, a multitude of end-to-end paths are provided, bringing multipath diversity to the network that could be explored by network coding techniques.

The AC-RLNC solution presented in this article is extremely helpful in realizing the throughput and delay requirements of the envisioned services in Smart Cities as considered recently in [3] for wireless backhaul solutions in 5G networks. The availability of multiple paths, using AC-RLNC to reach the desired throughput-delay tread-off, enhances the resilience of real-time applications like video streaming, vehicle to everything (V2X) communications, etc.

In this paper we propose an architecture that fosters the exchange of control information between network coding communication solution, SDN, network function virtualization (NFV) management, and orchestration components for general meshed heterogeneous communications and, as an propitious example, for services in Smart Cities powered by 5G wireless backhaul technologies.

The structure of this work is as follows. In Section II, we provide an overview of classical communication solutions over MS-MD networks. In Section III, we present the necessary background on AC-RLNC for single-path (SP), multipath (MP), and multipath and multi-hop (MP-MH) networks. In Section IV, we introduce the network architecture we propose for the heterogeneous MS-MD communications. In Sections V and VI, we present our scheme for communication over meshed MS-MD networks. In particular, Section V introduces the proposed scalable, secure, and efficient SDN Controller (SSE-SDNC), while Section VI introduces the generalized AC-RLNC in conjunction with the SSE-SDNC controller. In Section VII, we show a few important examples of enabling technologies and features. Table I lists the symbols that we use throughout this paper, along with their definitions.

II. Overview of Multiple Source Multiple Destination Networks

A. RLNC and MPTCP Communication

Transmission Control Protocol (TCP) and Internet Protocol (IP) were first proposed in the '1970s [4], [5]. The Classical TCP in the transport layer was first considered for a single point-to-point connection [6], [7], and the protocol was not capable of efficiently supporting the multipath communications as it relates applications to source and destination IP addresses and ports. Moreover, even for a single communication with a lossy link or with varying Round Trip Time (RTT) delay, TCP is known to be sub-optimal with major limitation on loss recovery time [8], [9].

Param	Definition
P, H	number of paths and number of hops
$\epsilon_{p,h}$	erasure probability of the p'th path and h'th hop
$r_{p,h}$	$1-\epsilon_{p,h}$, rate of the p'th path and h'th hop
t	time slot index
M	number of information packets
p_i	information packet, $i \in [1, M]$
$\mu_i \in \mathbb{F}_z$	random coefficients
$c(t,p)$	RLNC to transmit at time slot t on p'th path
k	number of packets in window, $RTT-1$
EW	end window of k new packets
w_{min}, w_{max}	limits of the effective window
w	length of the effective window
r_{G_p}	rate of p'th global path
$\bar{o}$	maximum window size
m_{G_p}	number of FECs for the p'th global path
a_{d_g}	number of added DoFs
m_{d_g}	number of missing DoFs
th	re-transmission margin
Δ	$P \cdot (d-1-th)$, re-transmission parameter
N_{new}	number of NEW-RLNC packets
N_{ret}	number of REP-RLNC packets

TABLE I: Symbol definitions

Multipath TCP (MPTCP) has been considered in the last decade, allowing multipath communication over naive selection of IP connections[1] between a source and a destination, for a single application using multiple TCP subflows simultaneously [10]. This multipath solution has been designed to be similar to regular TCP to cope with middleboxes. Fig. 2, top panel, illustrates the MPTCP solution, by simultaneously using multiple independent TCP subflows, over the naive multipath connections between the source and the destination. Although MPTCP provides better utilization of the available resources in a multipath communication, delay and loss variances result in packet reordering following on specific lost packets, increased out-of-order buffer, reduced overall throughput, which causing MPTCP solution at times performs worse than traditional TCP [11], [12].

Random Linear Network Coding (RLNC), was first introduced in the '2000s, achieving the min-cut max-flow of the multipath and multi-hop (MP-MH) networks by mixing long blocks of data with random linear coefficient over a large enough field [13]. The RLNC-coded packets transmitted across the multipath network are random linear combinations of the raw packets of information in a block [14]. The receiver, when obtains sufficient linear combinations, can decode all the information by performing a Gaussian elimination on a linear system [13], [15], [16]. Consider a sequence of M information packets $\{p_1, p_2, \dots p_M\}$ that need to be transmitted from a source to a destination through a network of interconnected nodes. Instead of sending the original packets, random linear combinations of these packets are transmitted, i.e., $\sum_{i=1}^{M} \mu_i p_i$ where the coefficients $\{\mu_1, \mu_2, \dots, \mu_M\}$ are randomly drawn at time step t.

Unlike MPTCP, to achieve the capacity of the MP-MH network, the raw data transmitted by the

[1]We refer to all the direct available connections between the source and the destination without considering the link rates and the bottleneck effects in the multi-hop communications as naive selection of global paths.

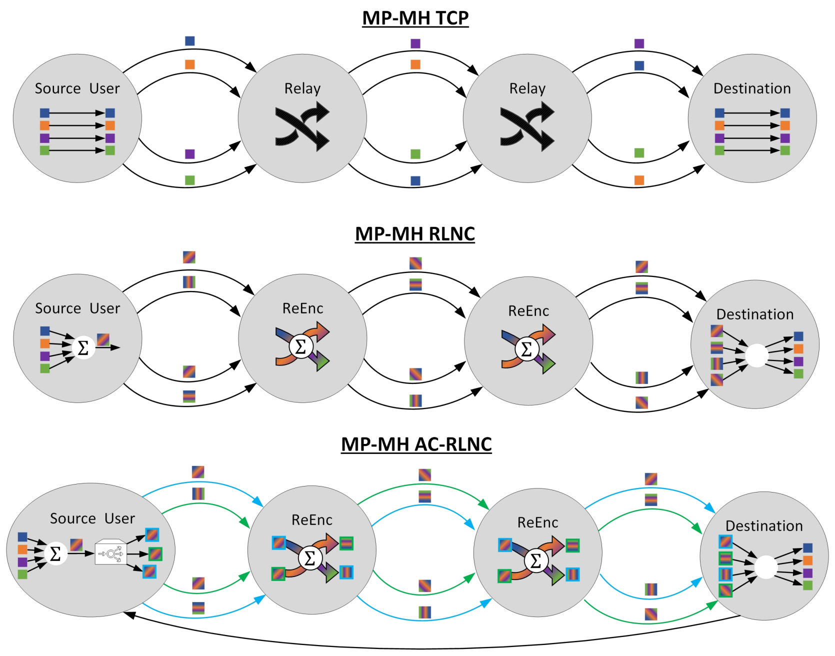

Fig. 2: For a MP-MH communication network: Top panel depicts traditional MPTCP communication solution; Middle panel depicts RLNC communication solution; Bottom panel depicts AC-RLNC communication solution.

source (in a block) is also mixed and re-encoded at the intermediate nodes[2] [13]. In a lossy multipath communication network with varying delay, the RLNC results in a robust solution in which the decoder, which decodes a block of linear combinations of the raw information, is not highly affected by the missing of specific packets. Fig. 2, middle panel, illustrates the RLNC solution mixing all the raw data at the source and re-mixing all the received data at the intermediate nodes using new random coefficients. Although RLNC in the MP-MH network may achieve the maximum throughput in the large blocklength regime, emerging advanced applications demand low in-order delivery delays while the high data rates require all the available resources of the network. Traditional information-theoretic solutions, including RLNC that requires large blocklength [17], are not able to reach this desired trade-off.

Recently, in single path communication using TCP and multipath communication using MPTCP, a priori forward error correction (FEC) according to the feedback acknowledgments was considered to reduce the in-order delay [18]–[21]. Coding techniques, e.g., MDS codes [22], [23] and systematic block codes [24], were considered as well. In particular, to compensate the average erasures in the lossy channels, RLNC is employed by periodically sending a priori FEC re-transmissions, i.e., the RLNC-coded packets generated from raw information in a sliding window block that has been used before for generating coded packets [25]–[27]. Coded TCP in single path proposes to send a different amount of repair symbols depending on the current loss rate. Although those solutions proposed recently in the literature are reactive to the feedback acknowledgments, namely causal, none of those solutions are tracking the varying channel condition and rate (not adaptive). This results in high delay or lower throughput that may be far from the desire reliability-delay trade-off.

B. SDN and Multiple Paths Communication

Multipath communication in conjunction with SDN has been used to enhance the performance and reliability in data centers and communication networks. When a node needs to communicate with

[2]In the re-encoding process proposed in the traditional RLNC, the intermediate nodes (so-called ReEnc) draw random coefficients for the packets coming from all the input links and prepare a new linear combination to transmit over all the output links. Note that decoding is not needed in the intermediate nodes.

another using MPTCP, through a multipath network, SDN can provide the information of the underlying infrastructure between the two nodes. The provided information (e.g. intermediate nodes) can be employed to determine the number of TCP connections that need to be created in order to leverage the availability of multipath network resources. Indeed, the combination of SDN and MPTCP leads to an efficient network resource utilization and congestion reduction, since the usage of idle and overloaded links are decreased. In addition, it also overcomes the challenges associated with traditional multipath routing approaches resulting from the Equal Cost Multipath (ECMP) algorithm, which randomly selects a path for the load balancing [28].

Other protocols, such as Stream Control Transmission Protocol (SCTP) and Quick UDP Internet Connections (QUIC) given in [29], [30] can also leverage the benefits of employing SDN, which allows them to establish a mapping between the routing information and the application streams. In fact, SDN controller can manage the configuration of devices (e.g. using OpenFlow, P4) to assign the transmission of packets from each stream over different paths [1].

SDN is also combined with segment routing approaches to decrease the delay in the transmission of packets and also to decrease the traffic volume in the distinct sub-networks [31]. Segment routing is a traffic engineering mechanisms that allows to use segment as an ordered list of instructions for packet routing.

III. Overview of Adaptive and Casual Random Linear Network Coding (AC-RLNC)

To achieve the desired delay-throughput trade-off demanded by the emerging advanced applications and technologies, i.e., low in-order delay and high data rate, recently proposed an Adaptive and Causal RLNC (AC-RLNC) solution for a communication with one source user and one destination over single path (SP) communication [32] and over MP-MH network [33]. In this section, we briefly review the AC-RLNC communication solution. AC-RLNC solution is adaptive to the link condition, and it is causal since the data transmissions from the source user depend on the particular erasure realizations, as reflected in the feedback acknowledgments from the destination. That is, AC-RLNC can track the erasure pattern of the links in the network and adaptively adjust re-transmission rates, using RLNC for a priori and posteriori FEC, based on the quality of the connections in the network, and is shown to narrows down the trade-off between high-throughput and low-delay. This section is important since the it is the base of the communication solution that will be presented for the MS-MD interconnected networks.

This solution was first introduced in [32], and it is both casual, as it is reactive to the feedback acknowledgements (received after one round trip time, i.e., RTT, delay), and adaptive, as the rate of re-transmissions is adapted based on the varying channel conditions, and thus this scheme is called adaptive and casual RLNC. The error correction mechanism is a combination of both forward error correction (FEC) and feedback FEC (FB-FEC). The FEC mechanism sends re-transmissions to compensate for erasures that are predicted in advance to reduce the delay, while FB-BEC sends re-transmissions based on the received feedback to improve the throughput. According to the actual link rates in the network, first, the source sends, a priori, an adaptive amount of FEC re-transmissions periodically. Then, at each transmission, according to posteriori re-transmission criterion, the source adaptively and causally decides over which paths to send FB-FEC re-transmissions and over which paths to send RLNC coded packets that contain new data information. The proposed re-transmission criterion is tracking the actual network packet receiving rate and the missing Degree of Freedom (DoF) rate required at the destination to decode the linear combination packets received as reflected from the feedback information.

We assume the forward channel from source to destination is erroneous, and the feedback channel is error-free. We aim at sending the coded packets such that the information packets can be decoded in-order and error-free at the destination while targeting a desired throughput and delay trade-off. In this section, we briefly review the AC-RLNC solution that was proposed for three different channel models in [32], [33]: single P2P (SP), multipath (MP), and multipath multi-hop (MP-MH) communication channels. The error-free feedback and the fixed RTT assumptions are considered for simplicity, the model can be

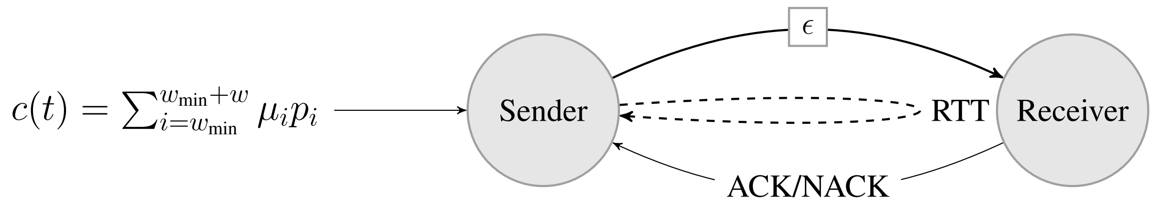

Fig. 3: System model and AC-RLNC encoding process for an SP communication channel.

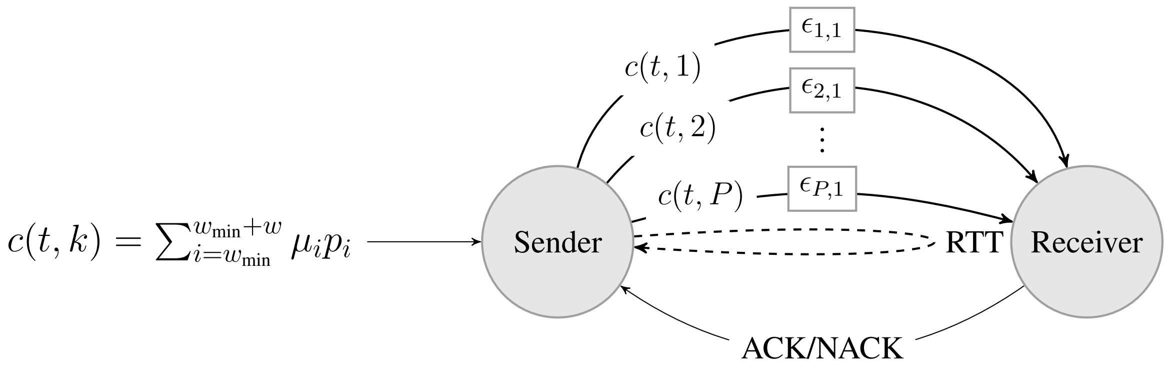

Fig. 4: System model and AC-RLNC encoding process for an MP communication channel.

extended for the case where there are errors in the feedback channel and RTT fluctuations e.g., by using the techniques provided in [34].

1) Single Point-to-Point Communication Channel (SP): Let consider a stream of information packets, i.e., $\{p_1, p_2 \dots\}$. At each time step t, RLNC is performed over a contiguous window, called *sliding window*, in the set of information packets, i.e, $c(t) = \sum_{i=w_{\min}}^{w_{\min}+w} \mu_i p_i$. The information packets involve in the linear coded packets are selected in sliding window mechanism to reduce delay and not in a large block of information as in traditional coding schemes, [13]. Moreover, at the AC-RLNC solution defined a maximum window size for the sliding window, denoted by $\bar{o}$, to limit the maximum new packets of information that may involve in the linear coded combinations, i.e., $w \leq \bar{o}$. This limit mechanism is used to bound the maximum delay. If the value of $w_{\max} = w_{\min} + w$ at time slots t and $(t-1)$ does not change, the packet $c(t)$ is called a re-transmission and denoted by REP-RLNC packet, otherwise, it is called a new-transmission and denoted by NEW-RLNC packet. In other words, $w_{\max}$ is incremented by each new-transmission, and $w_{\min}$ is incremented once the sender ensures that the decoder was able to decode each p_i, i.e., $i \leq w_{\min}$ (so-called in-order delivery).

Fig. 3 shows the system model and encoding idea of AC-RLNC in an SP network. The forward channel is considered a BEC(ϵ), where ϵ may change over the time, and the backward channel transmits error-free ACKs/NACKs that are received at the sender at time slot t for the packet that was transmitted at time $t-$RTT. The sender tracks the the channel state and decides whether to send a new-transmission or a re-transmission at each time step.

2) Multipath Communication Channel (MP): For the MP network, a new *adaptive discrete water-filling algorithm* is proposed at the source nodes for the allocation of the new coded packets of information and the re-transmissions over the available global paths [33]. This adaptive discrete water-filling algorithm balances between two objectives, maximizing throughput and minimizing the in-order delay, deciding the allocation of new and re-transmission packets to obtain the desired throughput-delay trade-off Fig. 2, bottom panel, illustrates the adaptive allocation scheme at the source nodes, when blue and green packets denote new and re-transmission packets of linear combination in the sliding window, respectively.

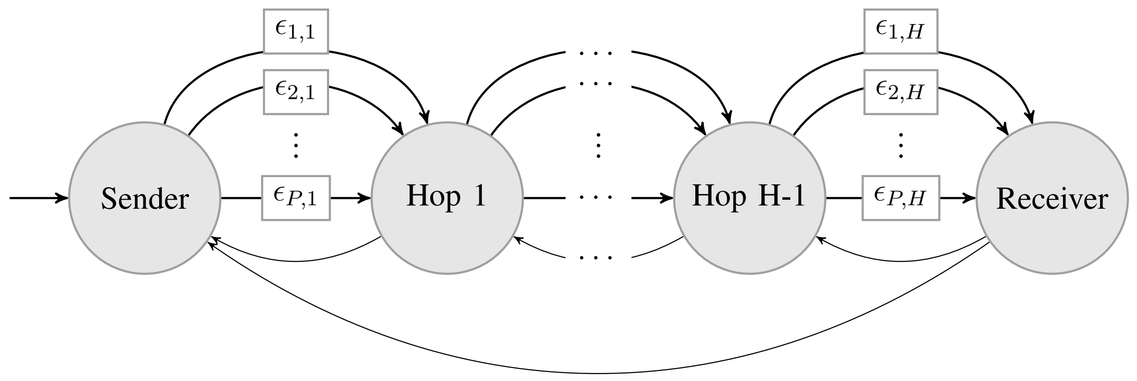

Fig. 5: System model for an MP-MH communication channel.

Fig. 4 shows the system model and encoding idea of AC-RLNC in an MP network. The forward channel is considered as composition of P parallel independent BEC channels with parameters $\{\epsilon_{1,1}, \dots, \epsilon_{P,1}\}$ (varying over time), and the backward channel transmits error-free ACKs/NACKs that are received at the sender at time slot t for the packets that were transmitted at time $t-\text{RTT}$. The sender tracks the channel state and determines a subset of paths for re-transmissions, and the rest of paths for new-transmissions. The process of assigning a new-transmission or a re-transmission to each path is done via a modified water-filling algorithm that takes into accounts both the throughput and in-order delivery delay targets in optimization.

3) Multipath Multi-Hop Communication Channel (MP-MH): The MP-MH channel is more general than MP channel and is depicted in Fig. 5. Each forward link can be considered as an independent BEC with parameter $\epsilon_{p,h}$ (varying over the time), where $p \in \{1, \dots, P\}$ and $h \in \{1, \dots, H\}$. The parameters P and $(H+1)$ are the number of paths between two consecutive hops and the number of hops (including the sender and the receiver), respectively. In Fig 5, it is assumed there are exactly P forward links between any two consecutive hops. In this document, we extend the model to the case where the number of forward links between two consecutive hops are not fixed. There is an error-free local feedback link between two consecutive hops, and an error-free global feedback link from the receiver to the sender. Each hop can estimate the erasure probabilities of its incoming and outgoing links based on the received forward messages and local feedback messages, respectively. In the AC-RLNC solution presented in [33], P global paths are determined from the sender to the receiver, and then the the AC-RLNC solution presented in Section III-2 is deployed over these P global paths to efficiently transmit NEW-RLNC packets over a subset of global paths and send REP-RLNC packets over the rest.

In the MP-MH networks, the rates in the naive connections between the source and the destination are limited by the link with the lowest rate. This causes a bottleneck if the rate of the constituent links of a path have dramatic variations. Hence, for the MP-MH networks in [33], is proposed a decentralized *balancing algorithm* that avoids the throughput degradation that may be caused by the bottleneck effects. This is obtained by reorganizing the selection of the naive global paths between the source and the destination to a new decentralized *global path* selection optimization at the intermediate nodes (so-called ReEnc nodes), considering the rates of the incoming and outgoing links at each ReEnc node, to reduce the bottleneck effects.

The process of randomly mixing the incoming RLNC packets to prepare another set of RLNC packets is called *traditional mixing*. By nature, all re-encoded packets generated in an intermediate node through traditional mixing are NEW-RLNC packets even if there was only one NEW-RLNC packet exist among the incoming packets to the node. Traditional mixing results in an increased in-order delivery delay. To increase the efficiency, in terms of maximizing throughput and minimizing in-order delivery delay, the ReEnc nodes perform a *selective mixing* in which the new-transmissions at each time step are re-encoded together and the received re-transmissions are also re-encoded together. In the bottom panel of Fig. 2), the links transmitting new-transmissions and re-transmissions using the selective mixing are marked in blue

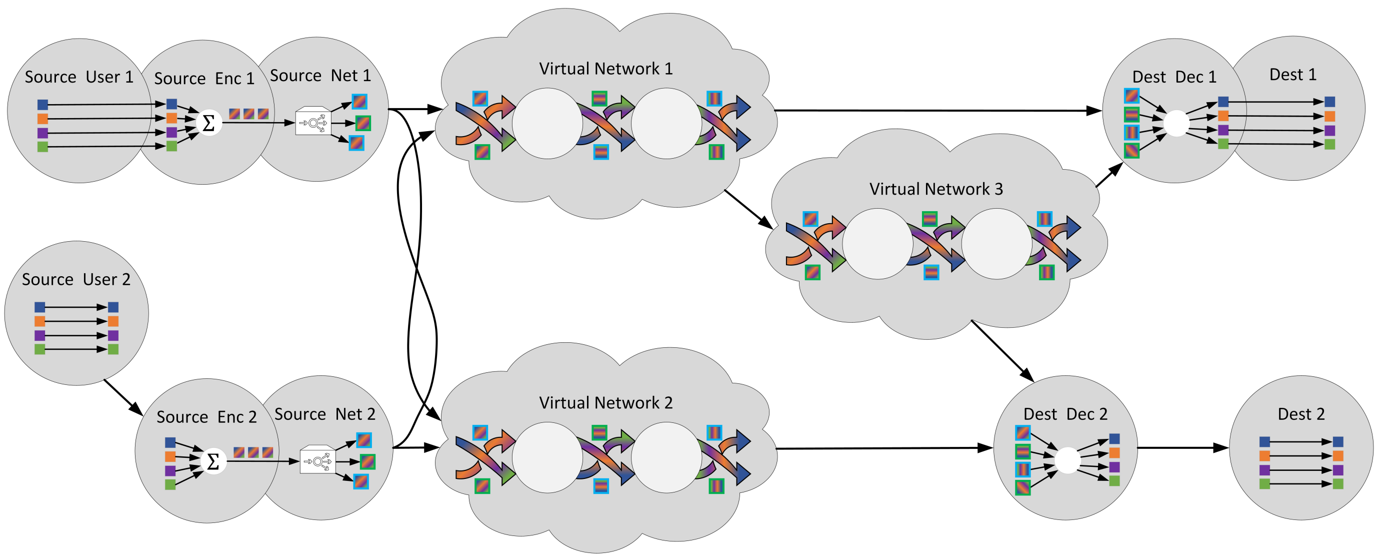

Fig. 6: A simple network architecture that is composed of three virtual networks, and has two sources and two destinations.

and green, respectively. This balancing algorithm together with the selective mixing solution closes the mean and max in-order delay gap and boosts the throughput. Thus, AC-RLNC for MP-MH network with one source and one destination can reach the desired delay-throughput trade-off.

For the non-asymptotic regime, AC-RLNC solution for MP-MH networks as proved in [33] may achieve more than 90% of the network capacity with zero error probability under mean and maximum in-order delay constraints. Fig. 2 demonstrates a comparison between TCP, traditional RLNC, and AC-RLNC communication communication solutions over MP-MH networs. We remark that in this paper, we also investigate the case of an MP-MH communication channel where a subset the intermediate nodes are not capable of performing AC-RLNC solution such as performing the re-encoding and balancing. This is an important feature since it makes the possibility of having a highly-heterogeneous network where a subset of nodes are not yet capable of adapting to the new technology, and consequently, provides a smooth transition phase. Incorporating the traditional nodes necessitate some modifications and considerations in the AC-RLNC solution that will be discussed in detail in Section IV.

The AC-RLNC solution proposed in [32] and [33] can reach the desired throughput-delay trade-off for an MP-MH network with one source and one destination, and showed to be very efficient in practice. However, modern communications have moved away from point-to-point models to increasingly heterogeneous highly-meshed networks with multiple sources and destinations that share the interconnected communication medium. Hence, to address the demands in the modern communications, in Sections VI-B and VI-C, we propose a novel controller-based protocol to demonstrate how it possible to deploy AC-RLNC solution in heterogeneous highly-meshed communication networks.

IV. Network Architecture for a Heterogeneous MS-MD communication

In this section, we present our new architecture for an MS-MD network of interconnected nodes. Each source intends to send messages to a subset of destinations through the shared network. We divide the network into several *Virtual Networks (VNs)*, where node arrangements are managed by the SSE-SDNC. Each VN has one communication channel to transfer the received packets, and the communication channel can be MP-MH (in the general case). We describe the main parts of the proposed network architecture through the example depicted in Fig. 6 , with two sources, two destinations, and three VNs. In this example, Source 1 can transmit information to both destinations, however, Source 2 can only transmit information to Destination 2. Each source is associated with one *User*, one *Enc node*, and one *Net node*, and each

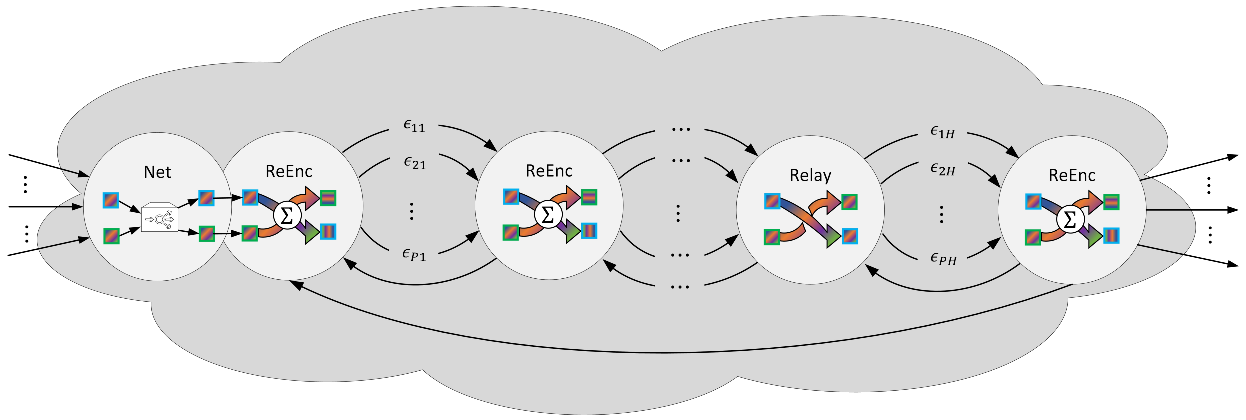

Fig. 7: The general architecture of a virtual network. When $H = 1$, the network topology we be reduced to an MP network. When $H = 1$ and $P = 1$, the network topology we be reduced to an SP network.

destination is associated with one *Dec node* and one *Dest node*. Each source is aware of the available paths and routing information for the current application to its desired destinations through the interaction with messaging systems, where the SSE-SDNC provides the updated values of the information regarding paths and links characteristics.

User generates the application/service information, and it can be located at the same place as the source nodes that generate the RLNC coded data, see Source User 1, or it can be physically apart, see Source User 2. In latter, User may deploy traditional communication methods to transmit the information to the first place on paths to its desired destination that supports the AC-RLNC solution, i.e., Enc and Net nodes. An Enc node prepares the RLNC-coded packets based on the received feedbacks and the tracked channel status. A Net node selects the global paths over which the new-transmissions are performed, called global paths of type-1, and subsequently, global paths over which the re-transmissions are transmitted, called global paths of type-2. Dec node is the closest to the destination that can perform decoding. The Dec node performs decoding, provides necessary feedback, and transmits the decoded packets to the destination. Finally, we use traditional communication protocol from Dec node to the destination. Dest node represent the physical location of the final destination.

We next introduce the main parts of a VN and their tasks. Fig. 7 depicts the system model for a VN in the general case of MP-MH topology. In a VN, *Net node* is responsible to select global-paths of type 1 and type 2. Then, other nodes are either a *ReEnc node* or a *Relay node*. A ReEnc node performs selective mixing (if supported) over the received RLNC packets and transmits the re-encoded RLNC packets on appropriate global-paths, or it performs traditional mixing (if supported) and transmits the re-encoded RLNC packets on global paths (all with same type). The Relay Nodes forward the received packets on their global paths whether they are re-transmissions or not using routing table.

V. Scalable, Secure and Efficient SDN Controller for Meshed MS-MD Communication

SDNs characterize network architecture by separating control plane from data plane and providing support for heterogeneous network interplay with rapid evolution and dynamism using programmable planes. Keeping up with rapid evolution and dynamism of network traffic SDN is a crucial element of 5G networks and beyond, along with other technologies like Network Function Virtualization (NFV). A SDN controller provides flexible management and control of network devices, thus it has the necessary information for the AC-RLNC solution over heterogeneous meshed networks with multiple sources and destinations. The Scalable, Secure and Efficient SDN Controller (SSE-SDNC) for mesh communications relies on multiple SDN controllers to attain an improved fault tolerance. The constituent controllers are organized in a cluster, so that the master node can be easily replaced by other slave nodes, as depicted in Fig. 8. The cluster of SDN controllers can be managed using available solutions like the Atomix ONOS

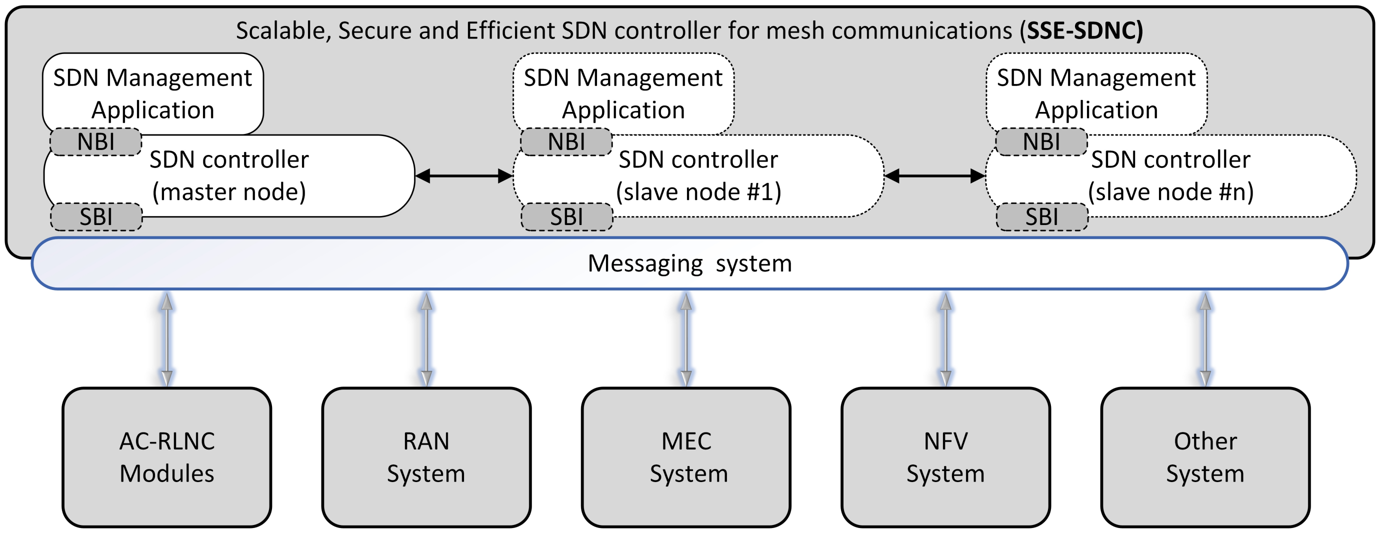
Scalable, Secure and Efficient SDN controller for mesh communications (**SSE-SDNC**)
SDN Management Application
NBI
SDN controller (master node)
SBI
SDN Management Application
NBI
SDN controller (slave node #1)
SBI
SDN Management Application
NBI
SDN controller (slave node #n)
SBI
Messaging system
AC-RLNC Modules
RAN System
MEC System
NFV System
Other System

Fig. 8: Scalable, Secure and Efficient SDN controller for mesh communications (SSE-SDNC) and messaging system.

cluster [35]. Within such approach the information regarding the network topology is distributed and accessible by the diverse controllers.

The SSE-SDNC, using multiple SDN controllers, manages the topology information of the virtual networks through the southbound interface (SBI), for instance, by supporting OpenFlow, P4, or NETCONF protocols. Such protocols allow to control the forwarding plane (e.g. flow rules in devices) and to manage devices (e.g. software configuration and updates). The SSE-SDNC also includes management application components that communicate with the SDN controller through the northbound API (NBI), supporting REST or gRPC technologies. The management application is responsible to retrieve the information from SDN controllers, regarding the network topology (e.g. RTT) and provide it to other components like the AC-RLNC modules. The information exchange between the SSE-SDNC and other components is performed through the messaging system, using YANG data models. The following subsections detail some of the solutions for each interface.

A. *SDN controllers for mesh networks in 5G*

The architecture of a SDN controller includes an application layer, responsible to interact with management applications (see section V-B), infrastructure layers, which is responsible to control and manage devices (see section V-C). A central layer includes the SDN controller with the management functionalities, and some controllers also include a Eastwestbound interface to allow the communication with other components or SDN controllers.

There are a plethora of SDN controllers available and the majority is released as open-source projects, as surveyed in [36], [37]. Some of these SDN controllers, worth mentioning include Ryu, OpenDayLight, ONOS and Floodlight. The main difference between them relies on the architecture (if running centralized, distributed or in a hierarchical fashion), the underlying programming language, the supported protocols in the distinct interfaces, the support for scalability, reliability and consistency. Given the requirements of the SSE-SDNC, Ryu and Floodlight are not suitable solutions due the centralized architecture model (i.e. introducing a single point of failure - SPOF), and do not scale to support large SDN deployments (e.g. networks in a smart city scenario) and have limited reliability support [38]. On the hand, OpenDaylight and ONOS are able to support distributed deployments and large SDN deployments, and have reliability support as required in the smart city scenario. In addition, the performance of these controllers does not diverge too much [39], [40]. For instance, OpenDayLight is able to provide a better performance regarding time to process flows, while ONOS is able to support higher flow response rate (flows/ms).

In a security perspective, either ONOS and OpenDayLight controllers have Denial of Service (DoS) and authentication vulnerabilities [41], with OpenDayLight presenting a higher number of vulnerabilities.

B. North Bound Interface - NBI

The North Bound Interface (NBI) of SDN controllers allow the bidirectional communication between controllers and management applications. Either for retrieving information regarding the status of network topology, as well as to configure the behaviour of the SDN controller when managing the flows tables in the networking devices (e.g. switches, routers).

The NBI commonly relies on REST interface, with the unidirectionality drawback. ONOS includes support for gRPC for bidirectional communications, which can be an advantage for SDN management applications, as those planned in the SSE-SDNC. Through the NBI, the SDN Management Application can configure the behaviour of the SDN controller, for instance by using available APIs, such as the intent or flow objective API, which allow to install flow rules in the devices by only requiring application to express their objective or intent (e.g. that a source node is able to communicate with a destination node). By expressing such intents, the SDN controller is able to install/manage the necessary flow rules in network devices.

C. South Bound Interface - SBI

The South Bound Interface (SBI) of SDN controllers is employed to manage, control networking devices such as switches, routers employing standard protocols like OpenFlow, P4 or NETCONF. The difference between OpenFlow and P4 is related with the fact that P4 is a programming language for the data plane in networking devices and Network Interface Cards (NICs) on end-nodes, as opposed to OpenFlow which allows the management of the forwarding and control data plane [42].

P4 introduces flexibility in the configuration of the devices, since the behaviour of common functionalities can be programmed with P4 language. For instance the forwarding behaviour in switches can be implemented in a P4 program, without requiring the explicit management of flow tables through the control plane like in OpenFlow. In this regards, several projects like the H2020 5Growth are employing P4 to program the 5G infrastructure to support end-to-end services [43]. Another advantage of employing P4 is related with the network telemetry, since the collection of information regarding networking devices can be included in the normal data packets, avoiding the necessity of using control messages to gather information [44]. Despite the increased flexibility, networking devices and NICs neet to run P4Runtime, which is responsible to run the P4 programs and to communicate with the SDN controllers [45].

The control and management of networking devices is also promoted with specific layers, implementing switch operating system, like stratum [46] which facilitates the management of devices in a scalable fashion.

D. Messaging System and Other Systems

The SSE-SDNC interacts with other systems and with the AC-RLNC modules through a messaging system, as illustrated in Fig. 8. The messaging can rely on publish subscribe (e.g. MQTT) or data stream models (e.g. kafka), which support communication between different systems in a distributed and reliable fashion.

In this regard the SDN management applications interacting the distributed SDN controllers can provide information regarding the network, events (e.g. new device available) through the messaging system, as a Kafka stream [47]. The advantage of relying in such kind of messaging is that available implementations of MQTT or Kafka provide fault tolerance by supporting multiple brokers to handle the diverse exchanged messages.

The information models exchanged through the messaging system will also comply with current standardization efforts [48]. The terminology and the diverse models specified in the ONF Common

Information Model (ONF-CIM), in the ONF Core Network Model (ONF-CNM), among others will be employed to further enhance the interoperability between SDN solutions and other systems. Such information models also include support for YANG, JSON data representation. The transport API (TAPI) [49] is also relevant to consider between the diverse systems.

VI. AC-RLNC in conjunction with controller for meshed MS-MD Communication

In this section, we present an AC-RLNC solution for the heterogeneous MS-MD network architecture. The solution was previously introduced in [32], [33], and in this article, we propose a practical implementation for an MD-SC network, where here some necessary modularization, modification, and interplay with main controller are incorporated, and thus, resulting in a decentralized implementation. This is a necessary step toward practical utilization of network coding in meshed heterogeneous networks as each module requires specific information and needs to be implemented in a specific part of the meshed network. We remind that the presented AC-RLNC solution is an example of how a distributed communication/computation solution can be deployed over a highly-meshed MS-MD network, by presenting effective architecture and protocols. One can use a similar approach to deploy other solutions, e.g., TCP communication, over the presented MS-MD network architecture.

A. *Main Modules of AC-RLNC Solution for MS-MD networks*

In this subsection, we describe the main modules of an AC-RLNC solution for the heterogeneous MS-MD network architecture, how they work together, and in which node in the proposed network architecture they need to be implemented. Then, we discuss the information each module requires to perform its tasks, and how this information is provided through interacting with the SSE-SDNC controller, communicating with other modules, and/or feedback. The main modules of the proposed AC-RLNC solution for heterogeneous MS-MD networks are the following:

- **Agent (AGN):** This module is present at all network nodes that require interaction with the controller, and is responsible to retrieve necessary information from SSE-SDNC, e.g., Fairness Table, RTT, Link Rates, through the messaging system, and/or to determine the information locally in the node like the Routing Table.
- **Balancing (BAL):** This modules matches the incoming links and outgoing links such that the bottleneck effect is minimized over each global path. The balancing module is implemented at ReEnc nodes.
- **Global Path Identification (GP):** The role of this module is identifying global paths between two parts of the network, and it is implemented at Net nodes in the proposed network architecture. In a Source Net node, the module identifies the available global paths for the current application/service from Source to Destination. In a VN Net node, it identifies the available global paths for the given application/service from the first to the last node in the VN.
- **Budgeting (BUG):** This module splits the rate budget (decides which global-paths are type 1 and which are type 2) for an application/service into re-transmissions and new-transmissions, and it is implemented at Net nodes.
- **Encoding (ENC):** This module performs RLNC encoding on the information packets of an application/service and prepares RLNC packets, and it is implemented at Source Enc nodes. We remind that each RLNC packet is either a re-transmission or a new-transmission.
- **Packet Allocation (PA):** The role of this module is allocating each global path of type 1 a new-transmission packet and each global path of type 2 a re-transmissin packet. This module is implemented at Source Net nodes and ReEnc nodes.
- **Re-encoding (REC):** This module linearly mixes the received RLNC packets and produces new RLNC packets, using selective mixing or traditional mixing. It is implemented at ReEnc nodes.

- **Decoding (DEC):** This module decodes the RLNC packets associated with the service/application that is desired by the destination and transmits necessary acknowledgment feedback messages. The decoding module is implemented at Dec node.

We describe the interaction between AC-RLNC modules and with the main controller in Section VI-B and VI-C, see also Fig. 9. Fig. 10 shows the OSI stack layers and nodes in which the modules are implemented. More details about the modules' implementations are given in Section VI-D. We note that if all or a subset of AC-RLNC modules cannot be run in parts of the network nodes, the controller can perform the task of the missing modules and make necessary changes according to the results.

B. Providing Necessary Information for Modules through SSE-SDNC

The SSE-SDNC is responsible for providing parts of necessary information for each module. Besides, the AGN module interacts with the SSE-SDNC to fetch the necessary information, and update necessary information. All information offered by the SSE-SDNC is not needed by every module in the presented solution, thus a set of distributed controllers, to have a low-delay module-controller communication, is also a promising idea - using the AGN modules, which communicates with the SSE-SDNC to synchronize updates in the network topology. The information that the AC-RLNC solution demands from the controller along with a brief explanation are listed below:

1) Routing Table (RT): Each RT is associated with a Net node, and it contains routing information (constituent links) between two IP addresses. In our solution, we need an RT per Net node. For example, an RT table that is requested by a Source Net node of an application/service contains all routing information from the source to the destination, however, an RT table that is requested by a VN Net node has all routing information from the first node to the last node in that VN. The AGN module is responsible for the RT information through interacting with the SSE-SDNC.

2) Fairness Table (FT): There exists one FT per Net node that shows how to split the available rate resources among several applications/services based on their priorities. For example, one can assign rate per application/service that is a real number in $[0, 1]$ such that these real numbers sum up to 1. One can also quantify the budget in several levels where the priority (urgency) of an application is quantified by a level. The SSE-SDNC is responsible for setting the weights/costs associated with the network resources. Then, AGN module presents this information to AC-RLNC modules at Net nodes.

3) Local-Path Routing Table (LPRT): There exists one LPRT per ReEnc node. An LPRT shows the matching between the incoming links and outgoing links, and it is constructed and updated by Balancing modules of a ReEnc node to minimize the bottleneck effect. The ReEnc nodes update their LPRTs based on dynamism of the network condition where the links quality may change over the time. Traditional Relay nodes do not need to update their LPRTs, and this provides heterogeneity property of using traditional nodes along with the nodes that are capable of performing the AC-RLNC solution. The AGN module is responsible for providing the RT information.

4) Global-Path Routing Table (GPRT): There is one GPRT per Net node of an application service. A GPRT includes all global paths routing information and rate between a pair of IP address, i.e., source and destination of an application/service when requested by a Source Net node, or first and last nodes of a VN when requested by a VN Net node. A GPRT is constructed and updated by Global path identification module with the assistance of the AGN module.

5) Round Trip Time (RTT) and Link Rates: The SSE-SDNC is responsible for providing updated values of the RTTs and link rates, which are determined through the probing mechanisms implemented in the SDN management applications.

Here, we briefly describe information that each AC-RLNC module requires (from the SSE-SDNC, and/or AGN or other modules) to perform its tasks. More details and algorithmic implementations can be found in Section VI-D. Fig. 9 illustrates a how different AC-RLNC modules and the controller exchange information during data transmission. The balancing module is called at a ReEnc node and requires the link rate for incoming and outgoing links (through the main controller, by probing, or from feedback), and

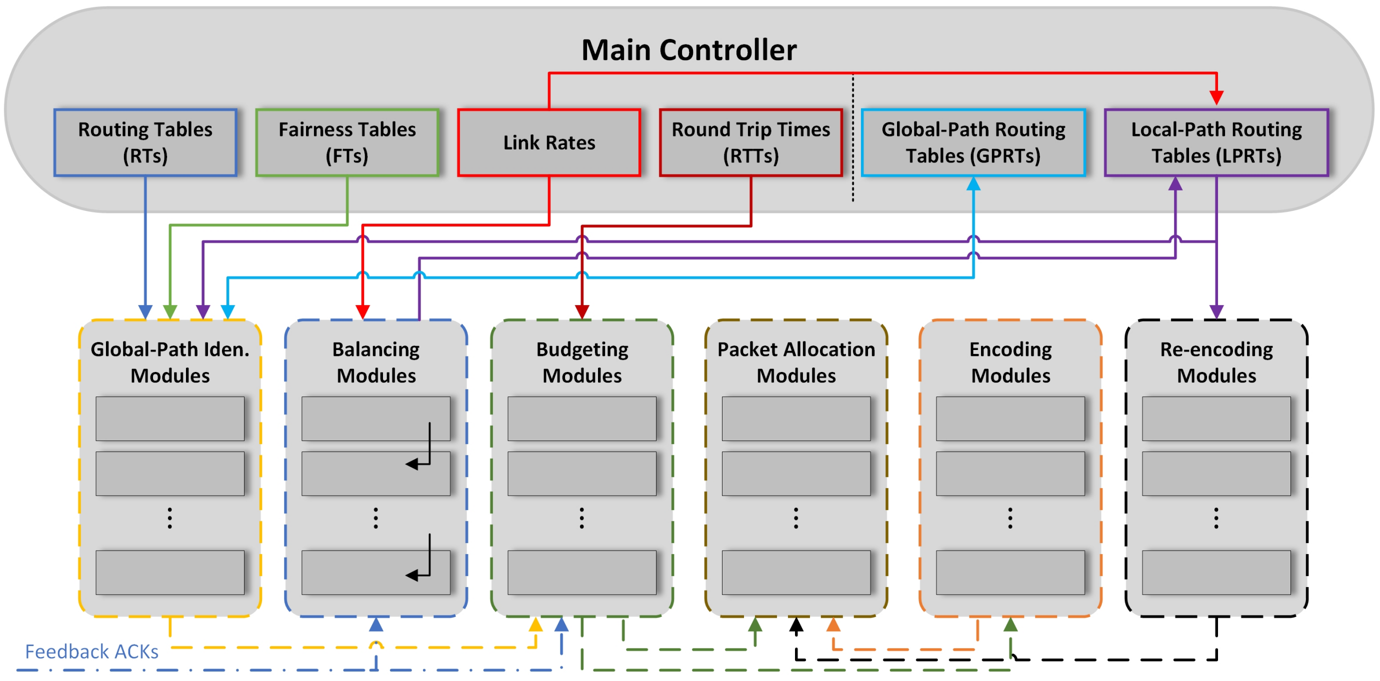

Fig. 9: The interaction of AC-RLNC modules and the main controller to realize the AC-RLNC solution in a heterogeneous MS-MD network. Note that the modules of one type, e.g., balancing modules, are implemented in different parts of the network, e.g., individual ReEnc Nodes. Here, for sake of presentation, we group all modules of same type together.

construct/update the LPRT of the node at SSE-SDNC, see Algorithm 3. For the global-path identification module, the required information depend on where it is called. At a Source Net node, the module requires RT, FT, and involved VN GPRTs to construct a new GPRT at the main controller. The GPRTs for the associated VNs (The VNs that carry information from Source to Destination) results in faster identification of global paths at a Source Net node. In a VN Net node, the module requires RT, FT, and LPRTs of constituent VN nodes, See Algorithm 4. The budgeting module requires the GPRT (from the global path identification module) and RTT (from the main controller, feedback, or through a probing mechanism), see Algorithm 5. The encoding module requires the number of new-transmissions and re-transmissions along with the sliding window limits from the budgeting module, see Algorithm 6. The re-encoding module performs re-encoding on the received RLNC packets based on the mixing mechanism (selective mixing, traditional mixing, or none) and prepares a new set of RLNC packets, see Algorithm 7. The decoding module performs decoding on received RLNC packets to retrieve the information packets when possible and also provides feedback.

Algorithm 1 shows how the SSE-SDNC initiates the AC-RLNC solution upon a new service/application request between a pair of User and Destination by collecting necessary information. Some initialization steps are performed by the SSE-SDNC (lines 1-15), and some are performed by AC-RLNC modules and the results are recorded in the SSE-SDNC (lines 16-31). After necessary initialisation steps, Algorithm 2 is called for the data transmission, line 32. Once the data transmission is done, necessary terminating steps are performed for the finished application/service request, lines 33-52.

The main controller performs the necessary initialisation in three stages. At first stage, the main controller checks if there already exists a routing table (RT) between User and Destination IP addresses, and if not, it constructs a new RT. Then, the RT is assigned to the new application/service (lines 1-5). Next, the main controller identifies the nodes that perform the AC-RLNC solution, i.e., Source Enc and Net nodes, VN Net nodes, ReEnc nodes, Relay nodes, and Dest Dec node involved in the current service/application using the RT (line 6). We remind that each application/service request needs one Enc node, one Source Net

Algorithm 1 Main AC-RLNC for SNOB 5G

```
    Called when a user requests an application/service from the Main-Controller
    Input: User and Destination IP addresses, Prioritization level of requested service/application (if supported).
    Initialization at the Main-Controller:
 1: Stage 1: Routing Table (RT)
 2: if no RT for (User, Destination) then
 3:     Build an RT for (User, Destination).
 4: end if
 5: Assign the RT to the new application/service.
 6: Specify the nodes involved in the new application/service and their roles given the RT.
 7: Stage 2: Fairness Table (FT)
 8: for each Net node involved with the new application/service do
 9:     if no FT assigned to the Net node then
10:         Construct an FT with the new application/service and assign it to the Net node.
11:     else
12:         Add the new application/service to the FT.
13:     end if
14:     Split the rate resources among the remaining applications/services (based on their priorities if supported).
15: end for
16: Stage 3: Provide the relevant information to AC-RLNC modules (see Fig. 9)
17: for each VN Net involved with the new application/service do
18:     for each application/service in the FT do
19:         Identify the global paths and their rates using Algorithm 4.
20:     end for
21: end for
22: for each VN Net involved with the new application/service do
23:     for each application/service in the FT do
24:         Match the incoming and outgoing links the nodes in the VN using Algorithm 3.
25:     end for
26: end for
27: for each Source Net node involved with the new application/service do
28:     for each application/service in the FT do
29:         Identify the global paths and their rates using Algorithm 4.
30:     end for
31: end for
    Transmission of the data
32: The data transmission algorithm is given in Algorithm 2.
    Termination at the Main-Controller:
33: for each Net node involved with the finished application/service do
34:     Remove the finished application/service from FT and .
35:     Split the rate resources among the remaining applications/services (based on their priorities if supported).
36: end for
37: Remove associated GPRTs and LPRTs from the main controller.
38: for each VN Net involved with the finished application/service do
39:     for each application/service in the FT do
40:         Identify the global paths and their rates using Algorithm 4.
41:     end for
42: end for
43: for each VN Net involved with the finished application/service do
44:     for each application/service in the FT do
45:         Match the incoming and outgoing links the nodes in the VN using Algorithm 3.
46:     end for
47: end for
48: for each Source Net node involved with the finished application/service do
49:     for each application/service in the FT do
50:         Identify the global paths and their rates using Algorithm 4.
51:     end for
52: end for
```

node, one or more VNs, and one Dec node. Per VN that carries information for the application/service, there are one Net node, and a number of Relay nodes and ReEnc nodes.

At second stage, the main controller adds the new application/service to the fairness table (FT) associated with each Net node that is involved with the new application/service, i.e., Net nodes that are on the routing paths of the new application/service. For the involved Net nodes that do not already have a FT, one with the new application/service is constructed. Then, the main controller splits the rate resources among the applications/services in each FT by updating priorities (lines 7-15). We note that a source can be assigned to more than one applications/services, and similarly a VN may carry information for more than one sources (consequently application/services). How to split the available rate resources (global paths) among the applications/services is determined in Net nodes using the FT. If the application/service prioritization is not supported in a Net node, the rate resources are split equally among the applications/services in the FT, including the new service/application. Otherwise, a discrete water-filling optimization can be deployed to distribute the global paths among the services/applications.

At third stage, the main controller calls the balancing modules and global-Path Identification modules to construct/update related GPRTs and LPRTs, lines 16-31. At this stage, one GPRT is constructed for the new application/service per Net node, and the remaining GPRTs associated with the involved Net nodes are updated accordingly. Similarly, one LPRT is constructed for the new application/service per VN node, and the remaining LPRTs associated with the involved VN nodes are updated accordingly. The main controller provides necessary information for the modules. This includes providing related RTs and FTs to global path identification modules, link rates to balancing modules, and RTTs to budgeting modules. First, for each involved VN Net node, the global path identification module is called for all applications/services in the FT (lines 17-21). Next, for each involved VN Net node, the balancing module is called for all applications/services in the FT (lines 22-26). Then, for each involved Source Net node, the global path identification module is called for all applications/services in the FT (lines 27-31).

Once the main controller finishes the initialization, Algorithm 2 is called to perform the application/service data transmission using the AC-RLNC solution (line 32), which will be describe in detail in Section VI-C.

Once the application/service data transmission ends, the main controller performs necessary termination steps as follows. First, the main controller removes the finished application/service from the fairness table (FT) associated with each Net node that is involved with the finished application/service. Then, the main controller splits the rate resources among the the remaining applications/services in each FT by updating priorities (lines 33-36). Next, the main controller removes the GPRTs (resp., LPRTs) associated with the involved Net nodes (resp., VN nodes) for the finished application/service (line 37). Finally, the main controller calls the balancing modules and global-Path Identification modules to update related GPRTs and LPRTs. First, for each involved VN Net node, the global path identification module is called for all remaining applications/services in the FT (lines 38-42). Next, for each involved VN Net node, the balancing module is called for all applications/services in the FT (lines 43-47). Then, for each involved Source Net node, the global path identification module is called for all applications/services in the FT (lines 48-52).

C. Cooperation of AR-RLNC modules to perform the complete solution

In this subsection, we precisely describe how the AC-RLNC modules collaborate and also interact with each other and with SSE-SDNC to perform the error-free data transmission task over a heterogeneous MS-MD network. Algorithm 2 shows how AC-RLNC modules perform AC-RLNC data transmission for an application/service altogether in a collaborative and distributed fashion. The operations performed at each node in the presented network architecture are described in more detail below:

- **User:** User needs to provide information packets related to the application/service to the source nodes (Source Enc and Net nodes). If User does not have the same IP address as the source nodes, it uses traditional methods to encode information packets into IP packets and transmit them to Enc Node (lines 2-4).

Algorithm 2 Main AC-RLNC for SNOB 5G - Data Transmission

```
    Algorithm is called by Main AC-RLNC Algorithm
 1: while information packets for the application/service to transmit do
    User
 2:     if User and Source nodes (Source Enc and Net nodes) do not the have same IP address then
 3:         Encode data in traditional IP packets and transmit them to Source Enc node using the RT.
 4:     end if
    Source Enc and Net nodes
 5:     Stage 1: Specify the global paths of type 1 (for new-transmissions) and the global paths of type 2 (for re-transmissions)
    using budgeting module given in Algorithm 5.
 6:     Stage 2: Prepare NEW-RLNC packets and REP-RLNC packets using encoding module given in Algorithm 6
 7:     Stage 3: Assign the RLNC packets through appropriate global paths using packet allocation module given in Algorithm 8.
    VN Net Node
 8:     Specify the global paths of type 1 (for new-transmissions) and the global paths of type 2 (for re-transmissions) using
    budgeting module given in Algorithm 5.
    ReEnc and Relay Nodes
 9:     for each VN Relay node do
10:         Forward coded packets using the RT.
11:     end for
12:     for each VN ReEnc node do
13:         Stage 1: Prepare NEW-RLNC and REP-RLNC packets using re-encoding module given in Algorithm 7.
14:         Stage 2: Assign the RLNC packets through appropriate global paths using packet allocation module given in
    Algorithm 8.
15:     end for
    Dec Node
16:     Stage 1: Decoding
17:     if there are sufficient RLNC packets for decoding then
18:         Decode the RLNC packets [50].
19:         if Dec node and Destination do not have the same IP address then
20:             Embed the decoded data into traditional IP packets and transmit them to Destination using the RT.
21:         end if
22:     else
23:         Update counter as number of RLNC packets received toward decoding the first information packet that is not
    decoded yet.
24:     end if
25:     Stage 2: Acknowledgment feedback
26:     if accumulative feedback is supported then
27:         Send back the index of the first information packet that is not decoded yet and its associated counter.
28:     else
29:         Send back ACKs for the received RLNC packets.
30:     end if
    There are changes in the network
31:     if link rates change then
32:         for each involved VN influenced by the link rate changes do
33:             Match the incoming and outgoing links of each node in the VN using Algorithm 3
34:             Identify the global paths and their rates for applications/services in the FT using Algorithm 4.
35:         end for
36:         for each involved Source Net node do
37:             Identify the global paths and their rates for applications/services in the FT using Algorithm 4.
38:         end for
39:     end if
40:     if a node leaves or a joins the network then
41:         Call Main-Controller to update the RT.
42:         For services with updated RTs and FTs use the initialization steps given in Algorithm 1.
43:     end if
44: end while
```

- **Source Enc and Net nodes:** Source nodes do their roles in the AC-RLNC solution in three stages: At stage 1 (line 5), the budgeting module for the current application/service is called to distribute the global paths between new-transmissions (type 1) and re-transmissions (type 2). The budgeting module performs a bit-filling optimization, using Algorithm 5, to perform this optimization. At stage 2 (line 6), the encoding module prepares the RLNC packet (a collection of re-transmissions and new transmissions). At stage 3 (line 7), the packet allocation module embeds each RLNC packet into an IP packet and assigns it to an appropriate global path (its corresponding IP address).
- **VN Net node:** Each VN Net node calls the budgeting module for the current application/service to distribute the global paths in the VN between new-transmissions (type 1) and re-transmissions (type 2), line 8.
- **ReEnc and Relay nodes:** For each VN Relay node, the incoming RLNC packets are forwarded on appropriate global paths using the RT (lines 9-11). VN ReEnc nodes do their roles in the AC-RLNC solution in two stages (lines 12-15): At stage 1, the re-encoding module is called to prepare NEW-RLNC packets and REP-RLNC packets. More detail on re-encoding module is given in Algorithm 7. At second stage, the packet allocation module embeds each RLNC packet into an IP packet and assigns it to an appropriate global path.
- **Dec node:** Dec node performs its tasks in two stages: decoding and transmission of feedback. At stage 1, lines 16-24, the decoding module is called that checks if sufficient number of RLNC packets are received to a recover an information packet. If true, the related RLNC packets are decoded, e.g., using Gaussian elimination method. Dec node needs to provide the retrieved information packet(s) to Destination. If IP addresses of Dec node and Destination are not the same, the information packets are encoded into IP packets and transmitted via traditional methods to Destination. When the RLNC packets received so far are not sufficient to recover new information packets, the Dec at this stage just records the number of received AC-RLNC packets that can help toward decoding first information packet that is not decoded yet (this index is denoted by w_{min}, and the recorded number is denoted by Dof. At Stage 2, lines 25-30, Dec node prepares and send the feedback messages. If cumulative feedback feature is supported, w_{min} and Dof are transmitted as feedback that convey the information about all packets that are decoded in-order at Destination and also the number of received RLNC packets toward decoding the first information packet that has not yet been decoded at Destination. In case the cumulative feedback feature is not supported, Dec Node sends an ACKs for all received RLNC packets.

At each time step, the SSE-SDNC checks if there are important changes in the network that affects the current application/service, e.g., changes in link rates, network nodes, etc, and performs necessary updates (lines 31-44). If link rates have notable changes, the following tasks are performed: the incoming and outgoing links of each node in the involved VNs are matched again using Algorithm 3. Then, the global paths and their rates for the involved Net nodes are identified again using Algorithm 4. For tracking link rate changes, a threshold function can be defined such that if the temporal link rate surpasses the threshold, the change is considered notable. For example, the threshold parameter can be set to 1.5 times the standard deviation of the link rate, and the temporal rate can be calculated during a specific period such as 3RTT time step. There is also a possibility that the architecture of the network changes during data transmission, i.e., an influential node joins or leaves the network. Then, the main controller needs to update the RT and perform the subsequent initialization using Algorithm 1.

D. Implementation of AC-RLNC Modules

In this subsection, we provide algorithm for each AC-RLNC module. Then, we present in which network layer of the Open Systems Interconnection model (OSI model) each module needs to be implemented based on the information the module needs and the nature of its algorithm. Fig. 10 shows the proposed network OSI stack layer implementation for the AC-RLNC solution.

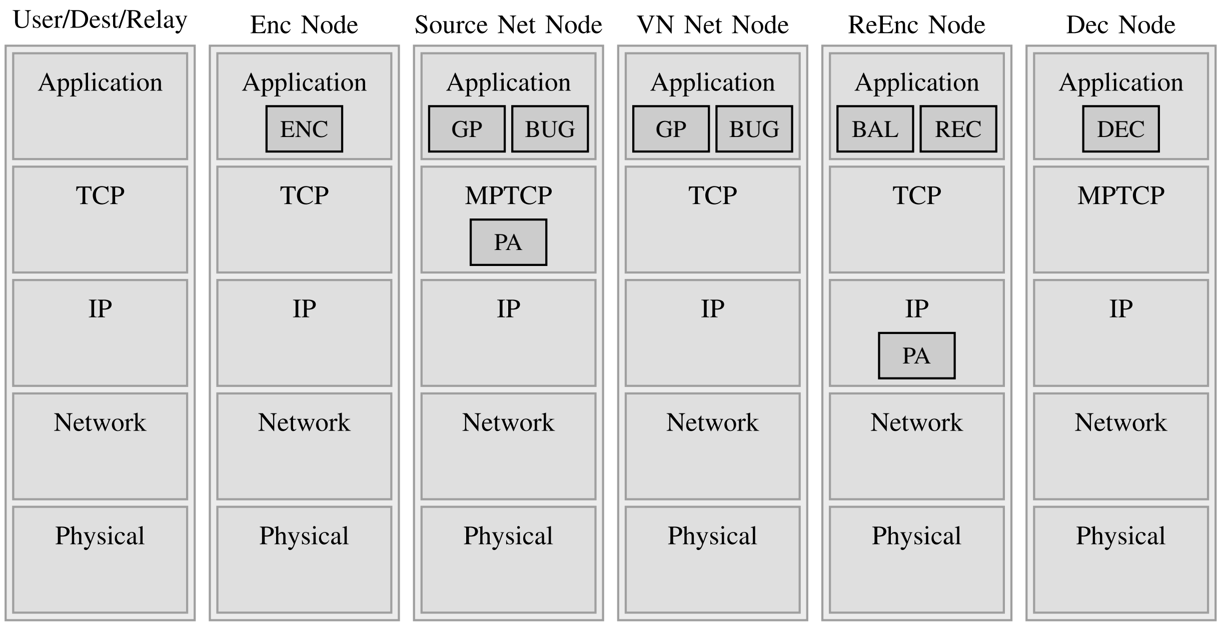

Fig. 10: Implementation of AC-RLNC modules.

1) Balancing Module Implementation: We remind that the role of balancing module is matching the incoming links and outgoing links of ReEnc nodes in a VN to improve the transmission throughput. The balancing module does this task by matching the incoming and outgoing links such that the link rate variations and correspondingly the bottleneck effect is minimized over VN global paths. For this purpose, a so called *natural matching* is defined in [33, Theoem 3] and briefly reviewed here: Consider the links between any two nodes in a VN are sorted and indexed in rate-decreasing order. Natural matching is matching the p'th incoming link with p'th outgoing link ($p \in \{1, \dots, P\}$ and P is the number of global-paths). It is shown in [33] that the natural matching is the optimal matching in terms of the resultant VN transmission throughput.

In the balancing procedure introduced in [33], it was assumed all hops (intermediate nodes) are capable of performing the balancing. However, in this paper, we consider a more general case where there can exist one or more Relay nodes, not capable of running the balancing module, between two consecutive ReEnc nodes in a VN. This extension which provides the heterogeneity property in the presented architecture necessities new considerations in the balancing module. We also assume the first node and the last node in each VN are ReEnc nodes. In matching the incoming links and outgoing links at a ReEnc node, the rate of each incoming and outgoing links are replaced with so called *associated rate*. The associated rate of an incoming link (resp., outgoing link) is summation of link rates on the global path that the link belongs to, starting from the previous ReEnc node (resp., ending to the next ReEnc node). In this way, the balancing module takes into account that the Relay nodes are not capable of performing link matching and makes more informed decisions.

Fig. 11 show an example of matching between input and output links in three different scenarios. The capacity between any two consecutive nodes is 2.6 in this example (summation of link rates between the two nodes). Each color represents one global path, and the rate of each global path is determined by the rate of the constituent link with lowest rate (marked in case the constituent links do not have a fixed rate and there is a bottleneck). The throughput then is summation of the rate of global paths. In scenario (A), a naive choice of global paths is considered because all intermediate nodes are Relay nodes and are not capable of optimal matching. The throughput in this scenario is 1.6. In scenario (B), a semi-optimal choice of global paths is considered. Here, one intermediate node is ReEnc node and capable of natural ordering of its incoming links. However, the second intermediate node is Relay node and is not capable

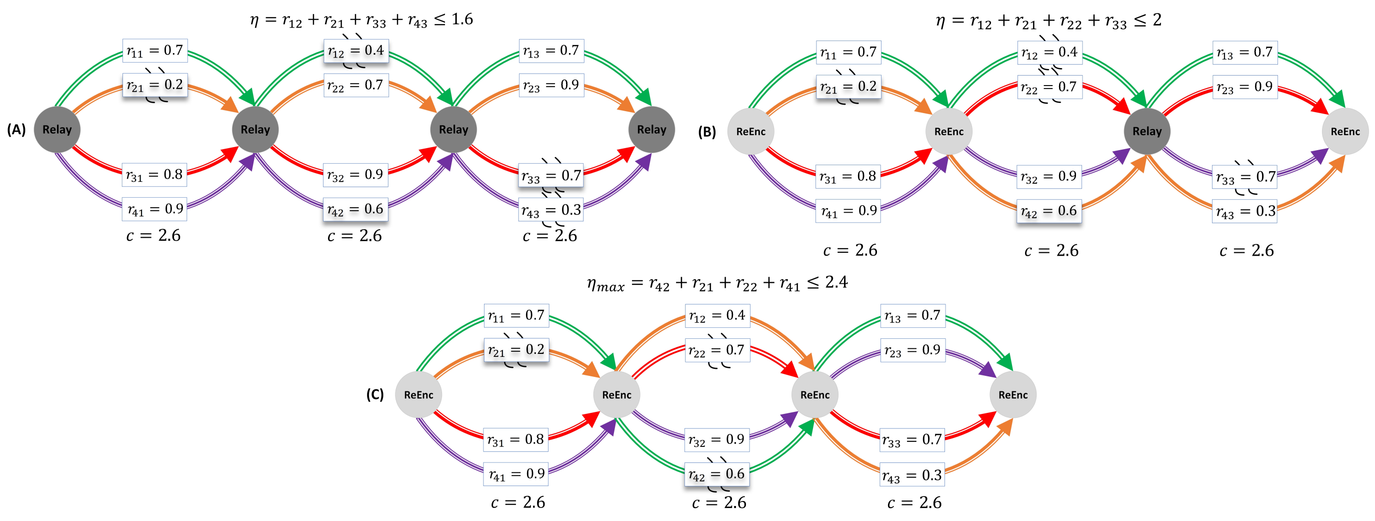

Fig. 11: Matching between input and output links in a VN. (A) shows the matching between input and output links for a naive choice of global paths. (B) shows the matching between input and output links when there is one intermediate node that is just Relay. (C) shows the matching between input and output links when all nodes are ReEnc nodes.

of natural ordering. Thus, the last ReEnc node performs the natural ordering of its incoming links using their associated rates to take into account that the previous Relay node only does a native matching. The throughput in this scenario is 2.0. In scenario (C), an optimal choice of global paths is considered. Here, all nodes are ReEnc nodes and capable of natural ordering of their incoming links. The throughput in this scenario is 2.4.

The balancing procedure can be implemented in a distributed fashion such that each ReEnc node in a VN performs the natural ordering of its incoming links based on their associated rates and sends the results to the next ReEnc node. To attain the associated rates for the incoming links, each ReEnc node requires the routing information and incoming/outgoing link rates of the relay nodes that appear between the previous ReEnc node and itself. When the natural ordering of incoming links are performed consecutively at each ReEnc node, in order they appear in the VN, and the results are sent to the next ReEnc node, each ReEnc is able to update the LPRT for the previous ReEnc node. If the Relay nodes do not supply feedback information to the next ReEnc node to estimate the associated link rates, the main controller needs to provide this information to the ReEnc nodes. In the initialization process in the Main controller, the balancing module at each ReEnc node in a VN requests the RT of previous Relay nodes (up to previous ReEnc node). In case of changes in the RT of the Relay node, the main controller inform the first ReEnc node after the Relay node about the change.

As algorithm 3 presents, the balancing module for a VN receives from the controller the incoming and outgoing associated rates for the current application/service. Then, for each ReEnc node, in order they appear in the VN (see Fig. 6), it finds the optimal matching and updates its LPRT accordingly. Because of the algorithmic nature of the balancing module, it is better to be implemented in application layer of each VN Re-Enc node.

Algorithm 3 Balancing module [33]

```
   Input: Input and output links' associated rates for each ReEnc node in a VN.
1: for each ReEnc node in order they appear in the VN do
2:     Match the incoming and outgoing links to the node using the natural matching.
3:     Send the matched order of the output links to the next ReEnc node in the VN.
4: end for
   Output: Update LPRT
```

2) Global-Path Identification Module Implementation: We remind that the role of global path identification module is to find all global paths along with their rates between two parts of the network architecture for an application/service. If the global path identification module is called by a Source Net node, the two parts are Source Enc and Dest Dec nodes. If the global path identification module is called by a VN Net node, the two parts are the first and last nodes in the VN.

Algorithm 4 is given for the global path identification module. The module called by a Source Net node uses the GPRTs associated with the involved VN Net nodes for a faster global path identification from Source to Destination. For a VN, a naive choice of global paths is attained by arbitrarily matching the input and output links of the constituent nodes. A naive choice of global-paths is used when a new application/service uses the VN. An optimal choice of global-paths is used for an existing application/service as LPRTs already exist for the ReEnc nodes. Then, the rate for each global path (r_{G_p} for p-'th global path, $p \in \{1, \ldots, P\}$ and P is the number of global paths) is computed and recorded in the GPRT. Global-path identification module is implemented in application layer of each Net node.

Algorithm 4 Global path identification module

```
    Input: RTs, FTs, GPRTs, LPRTs.
 1: if Called by a Source Net Node then
 2:     Create GPRT using GPRTs of associated VN Net nodes for the current application/service.
 3: else
 4:     if New service/application then
 5:         Create GPRT with naive choice of global paths.
 6:     else
 7:         Update GPRT with the updates LPRTs.
 8:     end if
 9: end if
10: Calculate the rate for each global path in GPRT.
    Output: Send rate of all global paths to the budgeting module as input.
```

3) Budgeting Module Implementation: This module is used in two types of nodes in the solution proposed, Source Net nodes and VN Net nodes. First, we will show the role of budgeting module at the Source Net nodes, and then we will explain the differences in the implementation of this module at the VN Net nodes. The role of budgeting module is to determine, at each time slot t, which global paths are of type 1, i.e., their number denoted by N_{new}, and which are of type 2, their number denoted by N_{ret}. At the Source Net nodes, it also keeps track of the effective window limits at each time step, i.e., w_{min} and w, requests the encoding module to prepare N_{new} NEW-RLNC packets and N_{ret} REP-RLNC packets at each time steps, and updates window limits accordingly. We note that the feedback is received from Dec node (resp., last ReEnc node) for the current application/service in the network (resp., the VN), and thus the budgeting module is aware of the received RLNC packets at the designated node after some delay.

We now review some necessary notations. The RLNC packet that is sent at time slot t over the p'th global path is denoted by $c(t, p)$, which is either a re-transmission or a new-transmission. The parameter w_{min} and w are the beginning and the length (in terms of the number of packets) of the effective window of the information packets. The parameter r_{G_p} denotes the rate of p'th global path and is updated during data transmission. The parameters m_{d_g} and a_{d_g} are the number of missing degrees of freedom and added degrees of freedom for decoding the information packets in the current effective window. The re-transmission criterion is defined as $\Delta > 0$, which will be described more in sequel. The parameter $\bar{o}$ is referred to maximum effective window size, i.e., $w \leq \bar{o}$. The parameter m_{G_p} denotes the number of FECs that are transmitted over p'th global path, per generation of k raw information packets (A reasonable choice is $k = RTT - 1$). The FEC packets are in fact a subset of re-transmissions that compensate in advance for the erasures that occur over a lossy channel regardless of the feedback[3]. For p'th global path and after transmitting first k new-transmissions, re-transmissions are sent in next m_{G_p} time slots, as FECs. The

[3]For example, when p'th global path can be modeled as a BEC with parameter ϵ_p, m_{G_p} can be set to $\lceil \epsilon_p k \rceil$.

remaining re-transmission packets that are not FEC and are sent in response to the received feedbacks are called FB-FECs.

At source Net nodes and at each time slot t, the budgeting module is run. First, it checks if feedback is available from Dec node. If no feedback is available yet (occurs in first k time steps or in case of feedback reception failure), all global paths are considered for new-transmissions (have type 1) as long as the end of generation of new packets (EW event) is not reached. If EW occurs and no feedback is received, all global paths are considered for re-transmissions (have type 2), lines 1-7. If feedback is available, the module makes necessary updates, lines 6-12. Based on the erasure patterns, the rate of each global path, i.e., r_{G_p}, is updated. Next, w_{min} is incremented by the number of new information packets that the decoding module has managed to decode in order at time slot $t - RTT$, and w using the updated w_{min} is updates as $w = w_{max} - w_{min}$. Then, the number of missing degrees of freedom m_{d_g} and the number of added degrees of freedom a_{d_g} are updated[4]. We remind that the parameters m_{d_g} is the number of new-transmissions that are erased, and a_{d_g} is the number of re-transmissions that are received at Dec node, for decoding the information packets in the current effective window. The budgeting module balances the new-transmissions and re-transmissions to hold $m_{d_g} < a_{d_g}$. The parameter Δ is used to provide a desired trade-off between the throughput and delay. For a desired trade-off between throughput and in-order delivery delay, a tune-able margin th is considered, and the re-transmission criterion is defined as $\Delta = P.(m_{d_g}/a_{d_g} - 1 - th) > 0$.

After necessary updates based on the received feedback, the budgeting module checks whether the maximum effective window size is reached. If true, all global paths are used for re-transmissions, lines 13-15. Otherwise, the budgeting modules effectively identifies the global paths that will carry new-transmissions and global paths that carry re-transmission. The parameter m_{G_p} shows the number of FECs that is due for the p'th global path as mentioned before. For FEC re-transmissions (lines 17-21), each global path p with $m_{G_p} > 0$ is selected for re-transmission, and N_{ret} and m_{G_p} are adjusted accordingly. For remaining global paths, the module identifies the global paths for FB-FEC re-transmissions, and selects the rest for new-transmissions. How the budgeting module does this split is based on a bit-filling optimization that maximizes the throughput of new information packets while minimizing the in-order delivery delay by providing sufficient re-transmissions. This optimization problem was introduced in [33], and we revisit it here in Proposition 1. In fact, if $\Delta > 0$, FB-FEC re-transmissions are needed and the appropriate global paths (of type 2) are selected according to the Proposition 1, and N_{ret} is adjusted accordingly, lines 13-27. Then, the remaining paths (if any) are considered for new-transmissions in case there are still new information packets to be added to the effective window, and N_{new} is adjusted, lines 28-33. When the end of generation of new packets is reached (EW event), the initialization for FEC re-transmissions are performed for all global paths, and also the unassigned global paths at the current time step are selected for FEC re-transmissions and N_{new} and their m_{G_p}s are adjusted accordingly, lines 34-41. Finally, the budgeting module requests N_{new} NEW-RLNC packets and N_{ret} REP-RLNC packets from the encoding module, updates w based on the new-transmissions occurred in the current time step, and sends the RLNC packets along with the global paths' types to the packet allocation module, lines 45-46. Algorithm 5 describes all the aforementioned steps.

For the bit-filling optimization solution as given in [33], we define the set of available global paths between the source to the destination as $\mathcal{P}$, and the index of a possible sub-set as $\xi \in \{1, \dots, 2^P\}$. We denote the global paths with indices in ξ as $\mathcal{P}_\xi$, and $\mathcal{P}_\xi^c = P \setminus \mathcal{P}_\xi$.

Proposition 1 (Bit-Filling [33])**.** *Given the estimated rates of available global paths between the source and the destination, r_{G_p} and $p \in \{1, \dots, P\}$, the source intends to maximize the throughput of the new packets of information. The set of paths $\mathcal{P}_{\hat{\xi}}$ on which new packets are sent, i.e., have type 1, is obtained as*

$$\hat{\xi} = \arg\max_{\xi} \sum_{i \in \mathcal{P}_\xi} r_{G_i}, \quad s.t. \quad \sum_{j \in \mathcal{P}_\xi^c} r_{G_j} \geq \Delta \tag{1}$$

[4]For an example of how to update the parameters m_{d_g} and a_{d_g}, refer to [33]

where the optimization problem minimizes the in-order delivery delay, by providing over the type 2 global paths a sufficient number of DoF's for decoding.

Now we elaborate on the role of the budgeting module at the VN Net nodes. The topology in each VN can be different from the general topology of the network. Namely, the number of global paths at the VNs can differ from the available global paths from source to destination. To increase the efficiency in the VNs, in terms of throughput-delay tread-off, this module determines at each time step t, out of the P global paths at the specific VN, which are of type 1 and which are of type 2. However, unlike the implementation of this module at the Source node, it is not required to manage the window size, as new raw information packets are not available at VN Net nodes. A new bit-filling optimization is given in Proposition 2 to determinate types of the global paths at VNs, and then this information is provided to the ReEnc node (see Section VI-D5) to utilize the different types of packets at the incoming links to re-encode and allocate them according to the determination of the budgeting module. In case there is no incoming NEW-RLNC packets but the bit-filling optimization results in $N_{\text{new}} > 0$, the module sets $N_{\text{ret}} = P$. Note that the budgeting module is used only when the selective mixing is supported in the VN.

We denote the set of available incoming paths to and constituent paths of the VN Net as $\mathcal{P}(s)$, where s distinguishes between incoming paths and outgoing paths respectively, i.e., $s \in \{In, Out\}$. We denote the index of a possible sub-set as $\xi(s) \in \{1, \dots, 2^{P(s)}\}$, and the paths with indices in $\xi(s)$ as $\mathcal{P}_\xi(s)$. Again, $\mathcal{P}^c_{\hat{\xi}}(s) = P(s) \setminus \mathcal{P}_\xi(s)$. Here, $\{p | p \in \mathcal{P}^c_{\hat{\xi}}(In)\}$ represents the incoming global paths to the VN that have type 2 (carry re-transmissions).

Proposition 2 (VN Net Bit-Filling). *Given the estimated rates of the incoming global paths to the VN, i.e., $\{r_{G_p}(In) | p \in \{$incoming global path$\}\}$, and the estimated rates of the VN global paths, i.e., $\{r_{G_p}(Out) | p \in \{$VN global path$\}\}$, the budgeting module intends to maximize the throughput of the NEW-RLNC packets in the VN. The set of paths $\mathcal{P}_{\hat{\xi}}(Out)$ on which NEW-RLNC packets are sent, i.e., have type 1, is obtained as*

$$\hat{\xi}(out) = \underset{\xi(Out)}{\arg\max} \sum_{i \in \mathcal{P}_\xi(Out)} r_{G_i}(Out), \quad \textit{s.t.} \quad \sum_{j \in \mathcal{P}^c_{\hat{\xi}}(Out)} r_{G_j}(Out) \geq \sum_{l \in \mathcal{P}^c_{\hat{\xi}}(In)} r_{G_l}(In) \tag{2}$$

where the optimization problem minimizes the in-order delivery delay, by providing over the type 2 global paths a sufficient number of DoF's at the VN.

The new bit-filling optimization in Proposition 2 provides the opportunity to handle the heterogeneity inside a VN network, e.g., having a variant number of paths between two consecutive intermediate nodes. In this case, a Net node needs to be associated with each ReEnc node (not just the first one as depicted in Fig 7) to perform the budgeting module (distribution of new-transmissions and re-transmissions along the outgoing links using the optimization described in Proposition 2) such that both throughput maximization and delivery delay minimization targets are satisfied.

4) Encoding Module Implementation: The role of encoding module is preparing the RLNC packets, Algorithm 6. The module first retrieves the application/service information packets from the received IP packets in case User and Source nodes do not have the same IP addresses (lines 1-3). An RLNC packet at time step t is formed by random linear combination of the application/service information packets in a properly defined sliding window. If for an RLNC packet $w_{max}(t) = w_{max}(t-1)$, it is a re-transmission and denoted by REP-RLNC. Otherwise ($w_{max}(t) > w_{max}(t-1)$), it is called a new-transmission and is denoted by NEW-RLNC. At time step t, the encoding module prepares N_{ret} Re-RLNC packets (lines 4-8) and N_{new} New-RLNC packets (lines 9-14) and send all packets to the packet allocation module as input. We note that $N_{\text{new}} + N_{\text{ret}} = P$ is number of global paths where N_{new} is the number of global paths of type 1 and N_{ret} is the number of global paths of type 2). Encoding module is implemented in application layer of each Source Enc node.

5) Re-Encoding Module Implementation: The role of re-encoding module is preparing new set of RLNC packets at ReEnc nodes, and is described in Algorithm 7. If selective mixing is supported, the incoming

Algorithm 5 Budgeting module at Source Net nodes [33]

```
    Input: Feedback acknowledgment, $r_{G_p}$, rates and unencoded RLNC data.
    Init: $N_{\text{new}} = 0$ and $N_{\text{ret}} = 0$
 1:
 2: if no feedback available then
 3:     if EW then
 4:         $N_{\text{ret}} = P$ and $N_{\text{new}} = 0$
 5:     else
 6:         $N_{\text{ret}} = 0$ and $N_{\text{new}} = P$
 7:     end if
 8: else
 9:     Update $r_{G_p}$ for each path
10:     Update $w_{min}$ and $w$
11:     Update $md_g$ and $ad_g$
12:     Update $\Delta$
13:     Size limit re-transmissions:
14:     if $w > \bar{o}$ then
15:         $N_{\text{ret}} = P$, $N_{\text{new}} = 0$
16:     else
17:         FEC re-transmissions:
18:         for each global-path $p$ with $m_{G_p} > 0$ do
19:             $N_{\text{ret}} = N_{\text{ret}} + 1$
20:             $m_{G_p} = m_{G_p} - 1$
21:         end for
22:         if $P - N_{\text{ret}} > 0$ then
23:             FB-FEC re-transmissions:
24:             if $\Delta > 0$ then
25:                 Determine FB-FEC paths (Proposition 1)
26:                 $N_{\text{ret}} = N_{\text{ret}} + \sum$ FB-FEC paths
27:             end if
28:             new-transmissions:
29:             for all remaining $P - N_{\text{ret}} - N_{\text{new}}$ paths do
30:                 if not EW then
31:                     $N_{\text{new}} = N_{\text{new}} + 1$
32:                 end if
33:             end for
34:             FEC re-transmissions (initialization):
35:             if EW then
36:                 Set $m_{G_p} := \lceil \epsilon_{G_p}(RTT - 1) \rfloor$
37:                 for all remaining $P - N_{\text{ret}} - N_{\text{new}}$ paths do
38:                     $N_{\text{ret}} = N_{\text{ret}} + 1$
39:                     $m_{G_p} = m_{G_p} - 1$
40:                 end for
41:             end if
42:         end if
43:     end if
44: end if
45: Request from Enc module $N_{\text{new}}$ NEW-RLNC and $N_{\text{ret}}$ REP-RLNC packets with $w_{min}$ and $w$
46: $w = w + N_{\text{new}}$
    Output: Send the RLNC packets to the Packet Allocation module
```

Algorithm 6 Encoding module [33]

```
   Input: $N_{\text{new}}$ and $N_{\text{ret}}$, $w_{min}(t)$ and $w(t)$
 1: if User and Source nodes (Source Enc and Net nodes) do not have same IP address then
 2:     Decode the IP packets received from the User.
 3: end if
 4: for $k \leftarrow 1$ to $N_{\text{ret}}$ do
 5:     Draw $w(t)$ random coefficients for encoding process, $\{\mu_1, \dots, \mu_{w(t)}\}$. [50].
 6:     Pick information packets $\{p_i | i \in \{w_{min}(t), \dots, w_{min}(t) + w(t)\}\}$.
 7:     Prepare re-transmissions: $c(t,k) = \sum_{i=w_{\min}(t)}^{w_{\min}(t)+w(t)} \mu_i p_i$
 8: end for
 9: for $k \leftarrow 1$ to $N_{\text{new}}$ do
10:     $w_k = w(t) + k$
11:     Draw $w_k$ random coefficients for encoding, $\{\mu_1, \dots, \mu_{w_k}\}$.
12:     Pick information packets $\{p_i | i \in \{w_{min}(t), \dots, w_{min}(t) + w_k\}\}$.
13:     Prepare new-transmissions: $c(t, k + N_{ret}) = \sum_{i=w_{\min}(t)}^{w_{\min}(t)+w_k} \mu_i p_i$
14: end for
   Output: $N_{\text{new}} + N_{\text{ret}}$ RLNC packets.
```

NEW-RLNC packets received at the current time step are mixed together (via re-encoding using a new set of random coefficients), line 3, and the incoming REP-RLNC packets received at current time step and previous time steps (though buffers) are mixed together, line 4. If traditional mixing is supported, the incoming RLNC packets received at current time step and previous time steps (though buffers) are all mixed together, line 6. If no mixing is supported, received RLNC packet from each incoming link is mixed with RLNC packets received from the same link at previous time steps, line 8. Re-encoding module is implemented in application layer of each VN Re-Enc node. If the NEW-RLNC packets (resp., REP-RLNC packets) packets received at a ReEnc node are fewer than the number of global paths of type 1 (resp., global paths of type 2), by re-encoding over the received NEW-RLNC packets (resp., REP-RLNC packets) the missing packets are compensated.

Algorithm 7 Re-encoding module [33]

```
   Input: Received RLNC packets (NEW-RLNC packets and REP-RLNC packets).
 1: Draw random coefficients for re-encoding process.
 2: if Selective-mixing is supported then
 3:     Re-encode received packets at time slot $t$ from global paths of type 1 (with new random coefficients).
 4:     Re-encode received packets at time slots $\leq t$ from global paths of type 2.
 5: else if Traditional mixing is supported then
 6:     Re-encode received packets at time slots $\leq t$ from all global paths.
 7: else
 8:     Re-encode independently received packets at time slots $\leq t$ from each global paths.
 9: end if
   Output: Re-encoded RLNC packets
```

6) Packet Allocation Module Implementation: The role of this module is embedding the given RLNC packets into IP packets. If an IP packet includes a NEW-RLNC packet, the module sends it through a global path of type 1 (lines 2-5); otherwise, if it includes a REP-RLNC packet, the module sends it through a global path of type 2 (lines 6-9). Packet Allocation module is implemented in network layer of Source Net nodes and VN Re-Enc nodes.

We propose standard formats for the TCP packets and the IP packets that support the AC-RLNC solution, see Fig. 12. The information packets for the AC-RLNC solution are in fact the TCP packets[5] [25]–[27]. Since the AC-RLNC decoding module is performed in Application layer, the source and destination TCP

[5]When the TCP header and the TCP paylod are not separated and are considered one information packet, the setup will be more robust against network imperfections.

Algorithm 8 Packet Allocation module

```
    Input: $P$ RLNC packets, GPRT, $w_{min}(t)$, $w_(t)$ and $\vec{\mu}$.
 1: for each RLNC packets do
 2:     if NEW-RLNC packet then
 3:         REP = 0.
 4:         Encode the RLNC packet in IP packet using the network information showed in Fig. 12.
 5:         Allocate the IP packet to global path of type 1
 6:     else
 7:         REP = 1.
 8:         Encode the RLNC packet in IP packet using the network information showed in Fig. 12.
 9:         Allocate the IP packet to global path of type 2
10:     end if
11: end for
    Output: Transmit $P$ IP packets through appropriate global paths
```

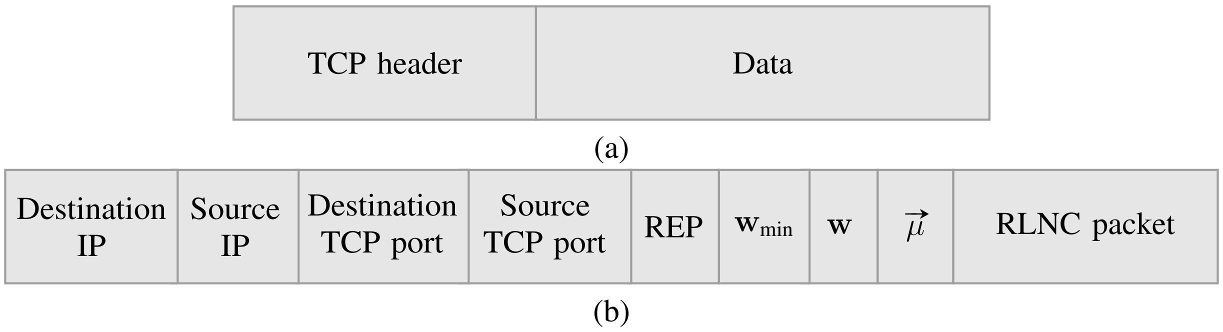

Fig. 12: Proposed (a) TCP packet structure, (b) IP packet structure.

ports along with their IP addresses are included in the IP packet. There is also a flag in the IP packet (REP flag) that shows if embeded RLNC packet is re-transmission (REP=1) or new-transmission (REP=0). The index (w_{min}) of first information packet and the number information packets (w) that is used for preparing the RLNC packet are included in the IP packet for re-encoding and decoding purposes. Lastly, the vector of w random coefficients is also embedded. We note that embedding the random coefficients in the IP packet is not necessary, and alternatively, one can embed the random seed that is used for generating the random coefficients [51], [52].

It is possible to have more than one RLNC packets in the payload of an IP packet as suggested in [25], [53]. The benefit of incorporating more than one coded packets in an IP packet is two-fold: First, RLNC coded packets with variable lengths can be suppurated, as they will be concatenated in one IP packet with a standard length. Second, having a short RLNC code length does not impede the network performance as multiple short RLNC coded packets form an IP packet with a standard length. Having multiple RLNC coded packet in one IP packet might need extra consideration in AC-RLNC modules.

VII. Visions and enabling technologies

Today's technological world demands effective distributed computation and storage to handle large volume of data with growing computational needs, security requirements, and delay sensitivities. The proposed scheme in this paper that brings network coding into SDN and provides an ultra reliable communication solution for heterogeneous networks, also opens a door for invaluable technologies and features that are beyond just reliable communications. Examples are but not limited to: security feature in controller and in data transmission, mmWave use case, and distributed computation and storage. The envisioning features and technologies are described briefly in this section.

A. Securing SDN controllers

Current SDN security challenges [54], [55] include proper authentication, access control, and data privacy and integrity among the orchestration components of SDN. In addition, (Distributed) Denial of Service attacks can be performed if centralized approaches for deploying SDN controllers are followed, for example due to limitations on management of flow tables that can be filled up by malicious/erroneous applications. Protection for other attacks like Man in the Middle (MITM) or replay attacks require enhancements in protocols like OpenFlow (to avoid changes in the fields' of control messages).

The SSE-SDNC herein proposed, tackles these security vulnerabilities by employing replication of controllers to avoid single point of failure (SPOF) and implements authentication, encryption and access control mechanisms to establish trust between the diverse components (e.g. SDN management applications, SDN controllers and network devices). For that, SSE-SDNC will perform replication of controllers with dynamic load distribution among controllers according to traffic conditions and load in the network, therefore addressing the SPOF issue. Authentication, access control and data privacy and integrity is achieved by integration with the Software-Defined Perimeter (SDP) security approach [54], that employs TLS mechanisms to enable mutual authentication and encryption of communications between clients (e.g. applications, network devices) and servers (e.g. controllers). In addition, SDP features an SDP control that is responsible by determining which devices/applications can connect to a given component (e.g. SDN controller), thus managing access control between clients and servers. The SSE-SDNC can be instrumented to interoperate with a SDP controller to only communicate with authenticated and trustworthy devices/applications.

So far in this subsection, the security aspects for the proposed controller are elaborated. It is essential to note that the utilization of network coding with post-quantum cryptosystems, as given in [56] for data transmission, results in a very efficient and promising security safeguards for the heterogeneous communications suggested herein. Combining AC-RLNC codes with a hybrid network coding cryptosystem presented in [56] enables a strong post-quantum security level with high communication rates. Surprising, this security level can be guarantee across all the links in the MS-MD network by encrypting a single link with computational cryptography.

B. mmWave mesh paths in AC-RLNC and SDN

The MP-MH AC-RLNC approach requires the knowledge of the throughput and delay of the several links and global paths, so that global paths of type 1 and type 2 can be determined. In a radio-based communication, the throughput and delay may depend on several different parameters beyond the link capacity and the data rate, since wireless links are very prone to interference and distance. The uncertainty on these global paths' parameters is increased if we consider the characteristics of mmWave, such as sensitivity to high co-channel interference, the lack of line of sight and the low propagation and penetration of the signal in outdoor environments. In these networks, links may suddenly disappear or their throughput may suddenly change. Therefore, the sudden and dynamic change in the throughput and delay will require a close interaction with the SDN controller, and determine not only new global paths, but also incorporating link stability in the choice of these paths.

To achieve the resiliency and agility needed at the mmWave backhaul, our approach is to collect additional metrics about the mmWave mesh network and link properties (i.e., packet loss, delay, RSSI, LQI, antenna direction [57]), in order to ensure faster recovery (in the presence of link failure), and optimized service selection. This information will be available to the SDN controller, and will be used to compute the link rates, in order to control the traffic flows inside the mesh network, optimizing the entire transport capacity of the mesh network between application services. With the metrics collected about the mmWave link state, the controller will be able not only to react to link failures, but also to predict before a link failure occurs, reconfiguring all the flows and forwarding tables according with the needs of the application services.

mmWave technology, on the other side, provides a great diversity on the establishment of different paths, achieving a multitude of end-to-end paths, and bringing multipath diversity to the network that is ready to be explored by the application of MP-MH AC-RLNC techniques. The configuration of the mmWave mesh can therefore be self-organizedly performed to accommodate the users and services needs over time, and optimize the overall network conditions, being it higher throughput on the links or optimal balancing to improve delay and service performance metrics.

C. Distributed storage and computation

Caching and distributed storage using network coding are demonstrated to be very effective in exploiting all available network resources and reducing the peak traffic overload of central units [58], [59]. Combining the caching and storage techniques, deployed in practice in today's modern networks, with the proposed adaptive and causal code proposed in this article, can satisfy the requirements of new applications that require ultra-reliable low-latency communications. As another enabling technology, we propose combining the idea of efficient distributed computation, and the adaptive and casual networking coding which results in ultra-reliable and low-delay distributed computations over a network of interconnected computational nodes.

A centralized computation scheme requires all data to be transmitted to a central unit for computation. However, with the enormous rate of data growth, the size of computational problems becomes larger and larger which results in huge communication and storage costs. Thus, the central computation scheme does not seem practical for large computational problems, e.g., multiplication of two large matrices, training a deep neural network, etc, and has led itself the viable and novel idea of distributed computation. In a distributed computation scheme, computations are performed at different nodes in the network which results in avoiding massive central computation, central storage, and unnecessary transmission of big chunks of data through a large parts of the network. In fact, computations at intermediate nodes can significantly reduce the communication and storage resource usage, and also reduce the process time per computational node. Distributed Computation has recently attracted significant attention. Recent interesting examples are, but not limited to, coded matrix multiplication [60], [61], where the multiplication of two large matrices are split to several small matrix multiplication tasks over computational nodes of a simple network topology; and [62] where a comprehensive information theoretical analysis is performed for achievable rates for distributed computation using network coding over a network with multiple sources and multiple destinations.

As an interesting envisioning technology, we propose combining the idea of distributed computation and the adaptive and casual networking coding which results in ultra-reliable and low-delay distributed computations over a network of interconnected computational nodes with unreliable communication links. The new MS-MD highly-meshed network architecture presented in this article provides the opportunity of performing sequence of computational jobs defined in-order at source nodes to be distributed among intermediate computational nodes. Then, each receiver aggregates enough chunks of computational results to identify the final computational results of the desired jobs in order. By borrowing introduced tools and ideas, e.g., local and global feedback, adaptation to the network changing dynamics, considering throughput-delay trade-off, etc, one can propose an effective adaptive and casual network coded computation over practical MS-MD networks.

VIII. Acknowledgments

The project SNOB-5G: Scalable Network Backhauling for 5G (reference CENTRO-01-0247-FEDER-045929) leading to this work is co-financed by the ERDF - European Regional Development Fund through the Operational Program for Competitiveness and Internationalisation - COMPETE 2020, the Center Portugal Regional Operational Program - CENTRO 2020 and the Portuguese Foundation for Science and Technology - FCT under the MIT Portugal Program.

REFERENCES

[1] Rezende, Pedro and Kianpisheh, Somayeh and Glitho, Roch and Madeira, Edmundo, "An SDN-Based Framework for Routing Multi-Streams Transport Traffic Over Multipath Networks," in *IEEE Int. Conf. Commun. (ICC)*. IEEE, may 2019, pp. 1–6.

[2] Z. Pi, J. Choi, and R. Heath, "Millimeter-wave gigabit broadband evolution toward 5G: fixed access and backhaul," *IEEE Communications Magazine*, vol. 54, no. 4, pp. 138–144, 2016.

[3] "SNOB-5G Scalable Network Backhauling for 5G." [Online]. Available: https://snob-5g.com/

[4] R. Kahn and V. Cerf, "A protocol for packet network intercommunication," *IEEE Transactions on Communications*, vol. 22, no. 5, pp. 637–648, 1974.

[5] V. Cerf, Y. Dalal, and C. Sunshine, "Specification of internet transmission control program," RFC 675, December, Tech. Rep., 1974.

[6] C. M. Kozierok, *The TCP/IP guide: a comprehensive, illustrated Internet protocols reference*. No Starch Press, 2005.

[7] K. R. Fall and W. R. Stevens, *TCP/IP illustrated, volume 1: The protocols*. addison-Wesley, 2011.

[8] J.-P. Martin-Flatin and S. Ravot, "Tcp congestion control in fast long-distance networks," *California Inst. Technol., Pasadena, CA, USA, Tech. Rep. CALT-68-2398*, 2002.

[9] I. Johansson, "Congestion control for 4g and 5g access," *Internet Engineering Task Force, Internet-Draft draft-johansson-cc-for-4g-5g-02*, 2016.

[10] C. Paasch and O. Bonaventure, "Multipath tcp," *Communications of the ACM*, vol. 57, no. 4, pp. 51–57, 2014.

[11] K. Yedugundla, S. Ferlin, T. Dreibholz, Ö. Alay, N. Kuhn, P. Hurtig, and A. Brunstrom, "Is multi-path transport suitable for latency sensitive traffic?" *Computer Networks*, vol. 105, pp. 1–21, 2016.

[12] S. Ferlin, Ö. Alay, O. Mehani, and R. Boreli, "BLEST: Blocking estimation-based MPTCP scheduler for heterogeneous networks," in *2016 IFIP Networking Conference (IFIP Networking) and Workshops*. IEEE, 2016, pp. 431–439.

[13] T. Ho, M. Médard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," *IEEE Trans. Inf. Theory*, vol. 52, no. 10, pp. 4413–4430, 2006.

[14] A. Schneuwly, D. Malak, and M. Médard, "Discrete water filling multi-path packet scheduling," in *2020 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2020, pp. 1658–1663.

[15] P. A. Chou, Y. Wu, and K. Jain, "Practical network coding," in *Proceedings of the annual Allerton conference on communication control and computing*, vol. 41, no. 1. The University; 1998, 2003, pp. 40–49.

[16] S. Patterson, "How MIT and Caltech's coding breakthrough could accelerate mobile network speeds'," *Network World*, 2014.

[17] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, D. A. Spielman, and V. Stemann, "Practical loss-resilient codes," in *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, 1997, pp. 150–159.

[18] A. Shokrollahi, "Raptor codes," *IEEE/ACM Transactions on Networking (TON)*, vol. 14, no. SI, pp. 2551–2567, 2006.

[19] M. Luby, "LT codes," in *Proc. IEEE Symp. Found. Computer Science*. IEEE, 2002, pp. 271–280.

[20] W. Guo, X. Shi, N. Cai, and M. Médard, "Localized dimension growth: A convolutional random network coding approach to managing memory and decoding delay," *IEEE Trans. Commun.*, vol. 61, no. 9, pp. 3894–3905, Sep. 2013.

[21] S. Ferlin, S. Kucera, H. Claussen, and Ö. Alay, "MPTCP meets FEC: Supporting latency-sensitive applications over heterogeneous networks," *IEEE/ACM Transactions on Networking*, vol. 26, no. 5, pp. 2005–2018, 2018.

[22] A. Badr, P. Patil, A. Khisti, W.-T. Tan, and J. Apostolopoulos, "Layered constructions for low-delay streaming codes," *IEEE Transactions on Information Theory*, vol. 63, no. 1, pp. 111–141, 2016.

[23] S. L. Fong, S. Emara, B. Li, A. Khisti, W.-T. Tan, X. Zhu, and J. Apostolopoulos, "Low-latency network-adaptive error control for interactive streaming," in *Proceedings of the 27th ACM International Conference on Multimedia*, 2019, pp. 438–446.

[24] J. Cloud, D. Leith, and M. Médard, "A coded generalization of selective repeat ARQ," in *2015 IEEE Conf. Computer Commun*, 2015, pp. 2155–2163.

[25] J. K. Sundararajan, D. Shah, M. Médard, S. Jakubczak, M. Mitzenmacher, and J. Barros, "Network coding meets TCP: Theory and implementation," *Proceedings of the IEEE*, vol. 99, no. 3, pp. 490–512, 2011.

[26] M. Kim, J. Cloud, A. ParandehGheibi, L. Urbina, K. Fouli, D. Leith, and M. Médard, "Network coded TCP (CTCP)," *arXiv preprint arXiv:1212.2291*, 2012.

[27] J. Cloud, F. du Pin Calmon, W. Zeng, G. Pau, L. M. Zeger, and M. Medard, "Multi-path TCP with network coding for mobile devices in heterogeneous networks," in *2013 IEEE 78th Vehicular Technology Conference (VTC Fall)*. IEEE, 2013, pp. 1–5.

[28] Lin, Ying Dar and Liu, Te Lung and Wang, Shun Hsien and Lai, Yuan Cheng, "Proactive multipath routing with a predictive mechanism in software-defined networks," *International Journal of Communication Systems*, vol. 32, no. 14, pp. 1–16, 2019.

[29] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. Iyengar *et al.*, "The quic transport protocol: Design and internet-scale deployment," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, 2017, pp. 183–196.

[30] Q. De Coninck, F. Michel, M. Piraux, F. Rochet, T. Given-Wilson, A. Legay, O. Pereira, and O. Bonaventure, "Pluginizing quic," in *Proceedings of the ACM Special Interest Group on Data Communication*, 2019, pp. 59–74.

[31] Pang, Junjie and Xu, Gaochao and Fu, Xiaodong, "SDN-Based Data Center Networking With Collaboration of Multipath TCP and Segment Routing," *IEEE Access*, vol. 5, pp. 9764–9773, 2017.

[32] A. Cohen, D. Malak, V. B. Brachay, and M. Medard, "Adaptive causal network coding with feedback," *IEEE Trans Commun.*, 2020.

[33] A. Cohen, G. Thiran, V. B. Bracha, and M. Médard, "Adaptive causal network coding with feedback for multipath multi-hop communications," in *IEEE Int. Conf. Commun. (ICC)*. IEEE, 2020, pp. 1–7. Under minor revision IEEE Trans Commun. (arXiv preprint arXiv:1910.13 290).

[34] D. Malak, M. Médard, and E. M. Yeh, "Tiny Codes for Guaranteeable Delay," *IEEE J. Sel. Areas in Commun.*, vol. 37, pp. 809–825, 2019.

[35] Xie, Junjie and Guo, Deke and Qian, Chen and Liu, Lei and Ren, Bangbang and Chen, Honghui, "Validation of Distributed SDN Control Plane Under Uncertain Failures," *IEEE/ACM Trans. Netw*, vol. 27, no. 3, pp. 1234–1247, jun 2019.

[36] Mamushiane, Lusani and Lysko, Albert and Dlamini, Sabelo, “A comparative evaluation of the performance of popular SDN controllers,” *IFIP Wireless Days*, vol. 2018-April, pp. 54–59, 2018.

[37] Amin, Rashid and Reisslein, Martin and Shah, Nadir, “Hybrid SDN networks: A survey of existing approaches,” *IEEE Communications Surveys and Tutorials*, vol. 20, no. 4, pp. 3259–3306, 2018.

[38] Bannour, Fetia and Souihi, Sami and Mellouk, Abdelhamid, “Distributed SDN Control : Survey , Taxonomy , and Challenges,” vol. 20, no. 1, pp. 333–354, 2018.

[39] Zhu, Liehuang and Karim, Md Monjurul and Sharif, Kashif and Li, Fan and Du, Xiaojiang and Guizani, Mohsen, “SDN Controllers: Benchmarking & Performance Evaluation,” pp. 1–14, Feb 2019. [Online]. Available: http://arxiv.org/abs/1902.04491

[40] Badotra, Sumit and Panda, Surya Narayan, “Evaluation and comparison of OpenDayLight and open networking operating system in software-defined networking,” *Cluster Computing*, vol. 0, 2019.

[41] Secci, Stefano and Diamanti, Alessio and Manuel Vilchez Sanchez, José and Tahirou Bah, Mamadou and Vizarreta, Petra and Mas Machuca, Carmen and Scott-Hayward, Sandra and Smith, Dylan, “Security and Performance Comparison of ONOS and ODL controllers,” ONF, Tech. Rep., 2019.

[42] Liu, Jed and Hallahan, William and Schlesinger, Cole and Sharif, Milad and Lee, Jeongkeun and Soulé, Robert and Wang, Han and Caşcaval, Călin and McKeown, Nick and Foster, Nate, “p4v: Practical Verification for Programmable Data Planes,” in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*. ACM, aug 2018, pp. 490–503.

[43] H. G. Project, “D2.2 : Initial implementation of 5G End-to-End Service Platform,” Tech. Rep., 2020.

[44] Guan, Bowei and Shen, Shan Hsiang, “FlowSpy: An efficient network monitoring framework using P4 in software-defined networks,” *IEEE Vehicular Technology Conference*, vol. 2019-September, 2019.

[45] The P4 Language Consortium, “P4 Language Specification,” https://p4.org/p4-spec/docs/P4-16-v1.2.1.html, 2020.

[46] Open Network Foundation, “Stratum: Enabling the era of next generation SDN,” https://www.opennetworking.org/stratum/, 2020.

[47] ONF, “kafka-onos,” https://github.com/opencord/kafka-onos, 2020.

[48] Janz, Christopher and Ong, Lyndon and Sethuraman, Karthik and Shukla, Vishnu, “Emerging transport SDN architecture and use cases,” *IEEE Communications Magazine*, vol. 54, no. 10, pp. 116–121, 2016.

[49] Open Network Foundation, “TAPI v2.1.3 Reference Implementation Agreement TR-547, Version 1.0,” July 2020.

[50] M. V. Pedersen, J. Heide, and F. H. Fitzek, “Kodo: An open and research oriented network coding library,” in *International Conference on Research in Networking*. Springer, 2011, pp. 145–152.

[51] “Sliding Window Random Linear code (RLC) Forward Erasure Correction (FEC) Schemes for FECFRAME.” [Online]. Available: https://tools.ietf.org/html/rfc8681

[52] “TinyMT32 Pseudorandom Number Generator (PRNG).” [Online]. Available: https://tools.ietf.org/html/rfc8682

[53] “Sliding Window Random Linear Code (RLC) Forward Erasure Correction (FEC) Schemes for QUIC.” [Online]. Available: https://tools.ietf.org/html/draft-roca-nwcrg-rlc-fec-scheme-for-quic-03#section-4.1.3

[54] A. Sallam, A. Refaey, and A. Shami, “On the Security of SDN: A Completed Secure and Scalable Framework Using the Software-Defined Perimeter,” *IEEE Access*, vol. 7, pp. 146 577–146 587, September 2019.

[55] S. Lee, J. Kim, S. Woo, C. Yoon, S. Scott-Hayward, V. Yegneswaran, P. Porras, and S. Shin, “A comprehensive security assessment framework for software-defined networks,” *Computers & Security*, vol. 91, April 2020.

[56] A. Cohen, R. D’Oliveira, S. Salamatian, and M. Médard, “Network Coding-Based Post-Quantum Cryptography,” *arXiv preprint arXiv:2009.01931*, 2020.

[57] M. Agiwal and A. Roy and N. Saxena, “Next generation 5G wireless networks: A comprehensive survey,” *IEEE Communication Surveys Tutorials*, vol. 18, no. 3, pp. 1617–1655, jun 2016.

[58] M. A. Maddah-Ali and U. Niesen, “Fundamental limits of caching,” *IEEE Transactions on Information Theory*, vol. 60, no. 5, pp. 2856–2867, 2014.

[59] A. G. Dimakis, K. Ramchandran, Y. Wu, and C. Suh, “A survey on network codes for distributed storage,” *Proceedings of the IEEE*, vol. 99, no. 3, pp. 476–489, 2011.

[60] U. Sheth, S. Dutta, M. Chaudhari, H. Jeong, Y. Yang, J. Kohonen, T. Roos, and P. Grover, “An application of storage-optimal matdot codes for coded matrix multiplication: Fast k-nearest neighbors estimation,” in *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 2018, pp. 1113–1120.

[61] S. Dutta, M. Fahim, F. Haddadpour, H. Jeong, V. Cadambe, and P. Grover, “On the optimal recovery threshold of coded matrix multiplication,” *IEEE Transactions on Information Theory*, vol. 66, no. 1, pp. 278–301, 2019.

[62] D. Malak, A. Cohen, and M. Médard, “How to distribute computation in networks,” in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 327–336.